Estimating Corpus Size via Queries

Andrei Broder[*] Marcus Fontura[*] Vanja Josifovski[*]
Ravi Kumar[*] Rajeev Motwani[†] Shubha Nabar[†]
Rina Panigrahy[†] Andrew Tomkins[*] Ying Xu[†]

[*]Yahoo! Research, 701 First Avenue, Sunnyvale, CA 94089.
{broder,marcusf,vanjaj,ravikumar,atomkins}@yahoo-inc.com
[†]Dept. of Computer Science, Stanford University, Stanford, CA 94305.
{rajeev,sunabar,rinap,xuying}@cs.stanford.edu

ABSTRACT

We consider the problem of estimating the size of a collection of documents using only a standard query interface. Our main idea is to construct an unbiased and low-variance estimator that can closely approximate the size of any set of documents defined by certain conditions, including that each document in the set must match at least one query from a uniformly sampleable query pool of known size, fixed in advance.

Using this basic estimator, we propose two approaches to estimating corpus size. The first approach requires a uniform random sample of documents from the corpus. The second approach avoids this notoriously difficult sample generation problem, and instead uses two fairly uncorrelated sets of terms as query pools; the accuracy of the second approach depends on the degree of correlation among the two sets of terms.

Experiments on a large TREC collection and on three major search engines demonstrates the effectiveness of our algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Measurements

Keywords

Corpus Size, Random Sampling, Estimator

1. INTRODUCTION

The overall quality of a web search engine is determined not only by the prowess of its ranking algorithm, but also by the caliber of its corpus, both in terms of comprehensiveness (e.g. coverage of topics, language, etc) and refinement (e.g. freshness, avoidance of spam, etc). Obviously, comprehensiveness is not the same as size — for instance as suggested in [8] one can easily build a public server for 100 billion pages each representing a random combination of Nostradamus prophecies and a dedicated search engine can as easily index them, in a feat of utter insignificance. See also `http://searchenginewatch.com/searchday/article.php/3527636` for a more serious discussion of comprehensiveness versus size.

Nevertheless, the ability to produce accurate measurements of the size of a web search engine's corpus, and of the size of slices of the corpus such as "all pages in Chinese indexed in Yahoo! from US-registered servers," is an important part of understanding the overall quality of a corpus. In the last ten years both the scientific literature and the popular press dealt at length with methodologies and estimates for the size of the various public web search engines and indirectly, the "size of the web". (See Section 2.)

Perhaps surprisingly, the problem of estimating the size of a corpus slice is also important for the owners of search engines themselves. For example, even the simple problem of efficiently counting the size of a result set raises non-trivial problems. In fact, Anagnostopoulos, Broder, and Carmel [1] show a Google example using the queries `george` and `washington` where the identity $|A \cup B| = |A| + |B| - |A \cap B|$ is off by 25%. (They present an algorithm that enables efficient sampling of search results, but to our knowledge, the required data structures have not been implemented in any web search engine.)

Further, the problems worsen when the goal is to estimate the *effective* size of a corpus slice; that is, the size of the slice discounted for pages that no longer exist, have been redirected, do not actually match the query, etc. In this case, even for a one term query, counting the number of documents in the term posting list gives only an upper bound of the effective size of the matching set.

An approach initiated by Bharat and Broder [6] is to produce only relative sizes, that is, to estimate the ratio between the sizes of several engines. Their approach is based on pseudo-uniform sampling from each engine index, first by sampling a query pool of English words, and then by sampling the results of the chosen query, followed by a capture-recapture estimates (that is, sample one engine, test containment in the other); however their method is heavily biased in favor of "content rich" documents and thus the ultimate results are problematic. This has been recently corrected by Bar-Yossef and Gurevich [4], who present a truly uniform sampling method that at least in principle should yield accurate ratios, but their method still needs to test containment, a non trivial procedure that introduces biases of its own. Note also that both the Broder and Bharat algorithm, as well as the Bar-Yossef and Gurevich algorithm yield only documents that match at least one query from their query pool, so in fact they only estimate the relative sizes of a certain subset of the corpus.

Estimating absolute corpus sizes appears to be a more difficult task. Previous work [13, 6] has had to fall back on sources other than public query interfaces, such as `http://searchenginewatch.com/reports/article.php/2156481` and `http://searchengineshowdown.com/stats`, or has had to trust the corpus size reported by search engines themselves. In this paper we propose a technique for estimating absolute corpus sizes without such additional sources.

1.1 Why corpus size matters

As we stated earlier, the ability to estimate the size of the corpus and of its various slices is important in understanding different characteristics and the overall quality of the corpus. A method for estimating absolute corpus size using a standard query interface can lead to methods for estimating the corpus freshness (i.e., fraction of up-to-date pages in the corpus), identifying over/under-represented topics in the corpus, measuring the prevalence of spam/trojans/viruses in the corpus, studying the comprehensiveness of the crawler that generated the corpus, and measuring the ability of the corpus to ably answer narrow-topic and rare queries [4]. Finally, relative corpus size estimates offers competitive marketing advantage and bragging rights in the context of the web search engines.

1.2 Our methods and results

Our method has two parts. First, we give a technique to count the size of specific broad subsets of the entire document corpus. Second, we show how to use this technique to generate an estimate of the size of the entire corpus.

1.2.1 Counting specific subsets

We begin our discussion of the method with an example. Assume that we wish to count the number of pages that contain an eight-digit number. A natural approach would be to produce a random sample of, say, 1 out of every 100,000 such numbers. One could then submit a query to a search engine for each number in the sample, count up the number of resulting documents, and multiply by 100,000. Unfortunately, some documents could contain many eight-digit numbers, and might therefore be counted multiple times by this procedure.

Let us modify the scheme as follows. Any document containing at least one eight-digit number should contribute 1 to the total count of documents in the set. But if a certain document d contains k different eight-digit numbers, we will allocate its total count of 1 by distributing $1/k$ to each of the eight-digit numbers it contains. In the previous scheme, when d is returned in response to a certain query, it contributes 1 to the overall count; in the new scheme it will contribute $1/k$, which is the reciprocal of the number of eight-digit numbers on the page.

Under the new scheme, we again take a sample of eight-digit numbers and submit each as a query. We then add up for each result document the reciprocal of the number of eight-digit numbers in that result document, and again multiply the final sum by 100,000. This new value is easily seen to be an unbiased estimator of the total number of documents; we provide a formal proof in Section 3.

More generally, our basic estimator allows us to count many different subsets D_A of the entire corpus D. The scheme will apply whenever D_A has two properties: first, a *query pool* A can be defined such that D_A is the union of the results of all queries in the pool A. And second, for any document, it is possible to determine efficiently how many queries from the query pool would have produced the document as a result.

We have seen eight-digit numbers as an example query pool. One could also employ queries chosen carefully from a query log. We will also consider approaches to modifying the query pool on the fly in order to reduce the variance of the estimator. Finally, the techniques also allow us to filter the results of each query on the fly, for example, by entirely removing documents that contain more than maximum threshold number of query terms from the pool, or that do not conform to a certain target length, specified as a range of bytes. The flexibility of the estimator is key to its power, as many issues that arise in the real world may be addressed by modifying the definition of D_A.

1.2.2 Employing the basic estimator

We have now defined a basic estimator to count the size of a subset D_A of the total corpus. This estimator may be used in many ways to perform counts of corpus sizes. The first method involves a random sample of the documents in a corpus, and an efficient function to determine whether a particular document belongs to D_A. Given this, we may simply estimate from the random sample the probability p_A that a document from the corpus belongs to D_A, and estimate the overall corpus size as $|D_A|/p_A$. We show in our experiments that this approach can provide very accurate measurements with quite reasonable sample sizes.

Nonetheless, it is quite difficult to generate a random sample of pages on the web, despite the efforts of many authors. Worse yet, when a technique is chosen and applied, there are no clean approaches to estimating the quality of the resulting sample, and so the sample bias cannot be understood. Even worse, because the problem of evaluating the quality of a random sample is so difficult, there has been very little academic work even to understand the extent to which techniques for generating random samples deviate from uniform.

Our basic estimator may also be applied in a second way to estimate the size of a corpus, without the presence of a random sample. We must resort to another assumption about the corpus, but one that is quite different from the assumption that a particular technique generates a uniform random sample. We assume that there are two query pools A and B that produce independent subsets D_A and D_B of the corpus. Note that independence here is different from disjointness. D_A and D_B may share documents, but the fraction of documents that belong to D_A should be the same whether we consider the entire corpus, or just D_B. If this is true, we may estimate the size of the corpus as $|D_A| \cdot |D_B|/|D_A \cap D_B|$.

The accuracy of this method depends heavily on the independence assumption, a crucial question in all probabilistic IR models. If the terms are correlated then we can only produce bounds based on their correlation. The following example might help build the reader's intuition for this issue. Assume that we apply our method using two sets of perfectly independent English terms and get a very accurate estimate of corpus size. Now the engine owners double its size by adding a large number of Chinese pages. If we repeat our experiments we will report the same number as before (since we will never or seldom see a Chinese page), even though the engine size has doubled. What happened? Well, our term sets used to be independent in the old corpus but now they are correlated: if we choose a page from D_A, it is now *more* likely to belong also to D_B just by dint of being in English.

This might seem as a limitation of our method, but in fact all query based estimation methods proposed so far suffer from this query vocabulary bias, and the contribution of our paper is to give a methodology where this bias can be correctly quantified by relating it to a fundamental concept in probabilistic information retrieval [16].

In some ways, the meaning of uncorrelated sets is a philosophical one. The estimator may be viewed as returning an estimate of that part of the corpus in which A and B are uncorrelated. While we do not advocate this perspective, we observe that if the sets are chosen appropriately, this may be the part of the engine of most interest from a measurement standpoint or it can be used to infer the relative sizes of search engines.

1.2.3 Experimental results

We first apply our methods to a large TREC collection consisting of over 1.2 million documents. Since we know the exact corpus size, these experiments are designed to demonstrate the effectiveness of our methods. We choose the query pool to be the set of all five-digit numbers. For the approach using random document samples, we obtain fairly high accuracies even with small number of samples. The error is significantly reduced when we modify the query pool to discard the most frequent terms in the query pool. We also estimate the corpus size using two uncorrelated query pools: set of five-digit numbers and a set of medium frequency words.

We then apply our methods to the Web by estimating what we call the "visible" corpus size of three major search engines. We choose the query pool to be the set of all eight-digit numbers and use this to estimate the visible corpus size. We also apply the two uncorrelated query pools approach: set of eight-digit numbers and a set of medium frequency words.

1.3 Organization

The paper is organized as follows. Section 2 discusses the related work on corpus size estimation and sampling random documents from a corpus. Section 3 presents the basic estimator and a variance reduction method. Section 4 presents our two approaches for corpus size estimation using the basic estimator. Section 5 contains the experimental results for a large TREC collection and Section 6 contains the experimental results for three major search engines. Section 7 concludes the paper.

2. RELATED WORK

Bharat and Broder [6] sampled the content of search engines using conjunctive and disjunctive queries composed of terms extracted from pages of the Yahoo! Directory. From a lexicon of 400K words, terms were combined to form around 35K queries. The sample consisted of one random URL chosen from the first 100 results returned by each queries. The fraction of the sampled pages from one search engine that are also present in the index of another search engine gives an estimate of the overlap between the two. The paper also estimates a search engine's coverage of the whole web from overlaps of pairs of search engines. A known problem with this technique is that it is biased toward content-rich pages with high rank.

The same year, Lawrence and Giles [13] reported size estimates using random queries based on a log of queries submitted to a search engine. To avoid bias toward highly ranked pages they use only queries for which the search engine retrieves all the results. However such query sampling does not provide a uniform random sample of the pages in the corpus. Later in [12], the authors extend the results and provide some other interesting statistics about search engine corpora and the accessible web. A big problem associated with directly comparing query results is that the approach is highly dependent on the underlying IR techniques used by the search engine to answer queries.

Guli and Signorini [10] repeated Bharat and Broder's experiments using Open Directory (`www.dmoz.org`) as source of keywords. They also used a modified technique to combine more than 2M terms into queries.

A core primitive in search engine size estimation is a way to obtain a uniform sample from its corpus using the query interface. Several papers report techniques to uniformly sample a search engine corpus using only the public query interface [3, 11, 13, 15, 4]. Many of them are based on random walks. A good survey of the methodologies and techniques used for the Web size and search engine size estimation is in [2].

Recently, Bar-Yossef and Gurevich [4] propose two new methods to obtain an unbiased sample of a search engine corpus. The first method is based on sampling, where queries are generated as word phrases from a corpus of documents and queries that return too many results are rejected. At a very high level, the analytic ideas in this method are similar to ours. The second method is based on random walks on documents and terms, but suitably unbiased using statistical techniques in order to produce a provably uniform document sample.

An interesting related issue is the study of mirrored hosts or duplicate pages and there has been much work done in this area; see, for instance, [5, 7]. Duplication is an important issue affecting the search quality of search engines, but the focus of this paper will be on size estimates.

Liu, Yu, and Meng [14] propose a method for estimating the corpus size based on the idea of two independent sets. Let D_1 and D_2 be two independent random sample of documents and let $D_3 = D_1 \cap D_2$. The search engine size is then estimated to be $|D_1||D_2|/|D_3|$. This idea is somewhat related to our method of uncorrelated query pools.

Callan and Connell [9] proposed query-based sampling in the context of distributed information retrieval for acquiring "resource descriptions" that accurately represent the contents of each database without relying on their internals.

This work is related to ours since we try to estimate corpus size based only on the search engine's public API. Wu, Gibb, and Crestani [17] propose methods for estimating and maintaining archive size information.

3. BASIC METHODS

In this section we discuss the basic estimator that will be the backbone of all our approaches. The mean of this unbiased estimator can be related to the corpus size. We assume that the index supports the basic query interface: given a query, return *all* the documents that match the query.

The first naive idea would be to use random queries and construct an estimator based on the number of documents returned for such queries. Unfortunately, this does not work, the difficulty being that there is no way of knowing the universe of all queries. Without this knowledge, it may not be possible to obtain an unbiased estimator.

We circumvent this difficulty by working with a *known* and *fixed* set of queries, called a *query pool*. For simplicity of exposition, we assume that each query is just one term. However, our methods will apply to any query pool that satisfies two conditions: first, the size of the pool should be known, and second, it should be possible to determine for any document how many queries in the pool match this document. We show (Lemma 1) how to construct an estimator whose mean is the number of documents in the corpus that match at least one query from this query pool.

Notation. Let D be the set of documents. We treat documents as a set of terms and use the notation "$a \in d$" to indicate that a term of query a occurs in the document d.

A query pool is a set of terms. For a query pool A, let $D_A \subseteq D$ be the set of documents such that every document in D_A contains at least one term in A. Define the *weight of a document* $d \in D_A$ with respect to A to be the inverse of the number of terms in A that occur in the document, i.e.,

$$w_d^A = \frac{1}{|d \cap A|}. \tag{1}$$

The definition of D_A guarantees that all weights are finite. The *weight of a query* a with respect to A is simply the weight of all documents containing the query, defined as follows:

$$w_a^A = \sum_{\substack{d \in D_A: \\ d \ni a}} w_d^A. \tag{2}$$

3.1 Basic estimator

Intuitively, if we encounter a document d which contains many query terms, each term should be given only partial credit for the document. Thus, we define our basic estimator as follows:

$$W_{A,D} = E_{a \sim A}[w_a^A], \tag{3}$$

i.e., the average weight of a query with respect to A.

We now show that this quantity times $|A|$ is an unbiased estimator of the number of documents containing at least one term in A.

LEMMA 1.

$$W_{A,D} = \frac{|D_A|}{|A|}.$$

PROOF.

$$\begin{aligned}
|A| \cdot W_{A,D} &\overset{(3)}{=} |A| \cdot E_{a \in A}\left[\sum_{\substack{d \in D_A: \\ d \ni a}} w_d^A\right] \\
&= \sum_{a \in A} \sum_{d \in D_A, d \ni a} w_d^A \\
&\overset{\text{swap}}{=} \sum_{d \in D_A} \sum_{a \in A, a \in d} w_d^A \\
&= \sum_{d \in D_A} w_d^A \left(\sum_{a \in A, a \in d} 1 \right) \\
&\overset{(2)}{=} \sum_{d \in D_A} 1 \\
&= |D_A|.
\end{aligned}$$

□

Thus, Lemma 1 guarantees an unbiased estimator whose mean is $|D_A|/|A|$. By sampling the query pool uniformly at random, $|D_A|$ can be estimated. We now discuss the issues in sampling.

3.2 Sampling

All the estimators such as $W_{A,D}$ can be estimated by the usual sampling techniques. We will illustrate the method to estimate $W_{A,D}$. Let X be the random variable given by $\sum_{d \in D_A, d \ni a} w_d^A$, where the query a is chosen uniformly at random from the query pool A. Clearly $E[X] = W_{A,D}$. We pick k terms $a_1, \ldots, a_k$ independently and uniformly at random from A and estimate the quantity $X_i = \sum_{d \in D_A, d \ni a_i} w_d^A$ for each of the a_i's. We then compute an *averaged estimator* $\overline{X}$ by averaging $X_1, \ldots, X_k$. It is easy to see that $E[\overline{X}] = E[X] = W_{A,D}$. Using Chebyshev's inequality, it follows that

$$\Pr[|\overline{X} - E[X]| \geq \epsilon E[X]] \leq \frac{1}{k}\left(\frac{\mathrm{var}[X]}{\epsilon^2 E^2[X]}\right).$$

Using this expression, if $k \geq (10/\epsilon^2)\mathrm{var}[X]/E^2[X]$ for instance, then with probability at least 0.1, the averaged estimator $\overline{X}$ approximates $W_{A,D}$ to within factor $(1 \pm \epsilon)$. To boost the probability of success from 0.1 to $1-\delta$ for an arbitrary δ, we can compute $O(\log 1/\delta)$ such averaged estimators and take their median value.

Ideally, we wish to make k as small as possible. To be able to do this, we need to make sure that $E[X]$ is not too small. For instance, this means that we cannot pick the query pool A to be terms that occur very rarely, since this will make the estimation of p_A harder. The second point to note is that if the variance var$[X]$ is large, then it implies that k has to be large. We address the second issue in greater detail in the next section.

3.3 Variance reduction

The variance of the random variable X can be very large. As an example, consider the following extreme scenario. The query pool A decomposes into A_1 and A_2 so that $|A_1| \ll |A_2|$ but each $a_1 \in A_1$ occurs in a large number of documents in D_A and each such document contains only terms in A_1. Consequently, the contribution to $W_{A,D}$ by $a_1 \in A_1$ is large. However, few random samples from A will hit A_1, owing to its small cardinality. Therefore, the number of samples need to be high. In other words, the distribution corresponding to X can have a heavy tail and we need a lot of samples to hit the tail (we give some evidence of this in Figure 1).

We now illustrate a generic method to ameliorate the problem: identify the tail and truncate it. Let $A' \subset A$ be those queries whose weights contribute to the tail of X of mass β; random sampling of documents in D, once again, can be used to identify candidate queries in A'. We then redefine a new query pool $\tilde{A}$ such that $\tilde{A} = A \setminus A'$. The hope is that the random variable $W_{\tilde{A},D}$ has lower variance than $W_{A,D}$. We now proceed to formalize this and analyze conditions under which truncation of a random variable causes its variance to reduce.

Notation. Let f be a probability distribution on an ordered domain U and let the random variable $X \sim f$. Consider the conditional random variable $Y = X \mid X < \tau$, i.e., its distribution $f|_{<\tau}$ is given by truncating f at τ and rescaling by the conditional mass $\Pr_{X\sim f}[X < \tau]$. We would like to study $\mathrm{var}[Y]$ vs. $\mathrm{var}[X]$.

If f can be arbitrary, then there can be no relationship between $\mathrm{var}[X]$ and $\mathrm{var}[Y]$. It is straightforward to construct an f such that $\mathrm{var}[Y] < \mathrm{var}[X]$. With little effort, we can also construct an f such that $\mathrm{var}[Y] > \mathrm{var}[X]$. Let f be a distribution with support of size three given by $f(-\epsilon) = f(\epsilon) = \delta/2$ and $f(1) = 1 - \delta$, for parameters $0 < \epsilon, \delta < 1$ to be specified later. Let $\tau = 1$ be the threshold. Let $X \sim f$ and let $Y = X \mid X < \tau$. Now, it is easy to see that $E[X] = 1 - \delta$ and $E[X^2] = (1 - \delta) + \delta\epsilon^2$ and so

$$\mathrm{var}[X] = (1 - \delta) + \delta\epsilon^2 - (1 - \delta)^2 = \delta(1 - \delta + \epsilon^2).$$

On the other hand, $E[Y] = 0$ and $\mathrm{var}[Y] = E[Y^2] = \epsilon^2$. Hence, if $\epsilon > \sqrt{\delta}$, we can achieve $\mathrm{var}[Y] > \mathrm{var}[X]$.

However, if f is monotonically non-increasing, then we show that $\mathrm{var}[Y] \le \mathrm{var}[X]$, i.e., truncation helps to reduce variance. In fact, in the extreme case, truncation can turn infinite variance into finite variance. When the distribution is a power law, we show a quantitative bound of the reduction in variance.

3.3.1 Monotone distributions

For simplicity, let us assume f is a discrete monotonically non-increasing distribution. Without loss of generality, let the support of f be $[n]$ with $f(1) \ge \cdots \ge f(n)$ and without loss of generality, let $\tau = n$. Let $g = f|_{<\tau}$, and $X \sim f, Y \sim g$. Notice that for $i = 1, \ldots, n-1$, $g(i) = f(i)/(1 - f(n))$. Let $\mu = E[f]$.

LEMMA 2. $f(n)(n-\mu)^2 \ge \sum_{i=1}^{n-1}(g(i) - f(i))(i-\mu)^2$.

PROOF. First, we show that for $1 \le i \le n$,

$$(n - \mu)^2 \ge (i - \mu)^2, \qquad (4)$$

or equivalently, $\mu \le (1 + n)/2$. Without loss of generality, assume n is odd and let $n' = n/2$. Let $a_1 = \min_{i<n'} f(i) = f(\lfloor n' \rfloor)$ and let $a_2 = \max_{i>n'} f(i) = f(\lceil n' \rceil)$. Thus, $a_1 \ge a_2$. Observe that $\mu = n' - \sum_i (n' - i)f(i)$ and

$$\begin{aligned}\sum_i (n'-i)f(i) &= \sum_{i>n'}(n'-i)f(i) - \sum_{i<n'}(i-n')f(i)\\ &\ge \sum_{i>n'}(n'-i)a_2 - \sum_{i\le n'}(i-n')a_1\\ &= (a_2 - a_1)\sum_{i<n'}(n'-i)\\ &\ge 0,\end{aligned}$$

and thus, $\mu \le n' = n/2$, establishing (4). Finally,

$$\begin{aligned}&\sum_{i=1}^{n-1}(g(i)-f(i))(i-\mu)^2\\ &\overset{(4)}{\le} \sum_{i=1}^{n-1}(g(i)-f(i))(n-\mu)^2\\ &= (n-\mu)^2\sum_{i=1}^{n-1}(g(i)-f(i))\\ &= (n-\mu)^2 f(n).\end{aligned}$$

□

LEMMA 3. $\mathrm{var}[Y] \le \mathrm{var}[X]$.

PROOF.

$$\begin{aligned}\mathrm{var}[X] &= \sum_{i=1}^{n} f(i)(i-\mu)^2\\ &= \left(\sum_{i=1}^{n-1} f(i)(i-\mu)^2\right) + f(n)(n-\mu)^2\\ &\overset{\text{Lemma 2}}{\ge} \sum_{i=1}^{n-1} f(i)(i-\mu)^2 + \sum_{i=1}^{n-1}(g(i)-f(i))(i-\mu)^2\\ &= \sum_{i=1}^{n-1} g(i)(i-\mu)^2\\ &\overset{(*)}{\ge} \sum_{i=1}^{n-1} g(i)(i-E[g])^2\\ &= \mathrm{var}[Y],\end{aligned}$$

where (*) follows since the convex function $\sum_i g(i)(i-y)^2$ is minimized at $y = E[g]$. □

3.3.2 Power law distributions

We consider the important case when f is given by a power law. In this case we obtain a quantitative bound on the reduction of the variance. For simplicity, we assume that f is a continuous distribution defined on $[1, \infty)$. Let $f(x) = \Pr[X = x] = \alpha x^{-\alpha-1}$ for some $\alpha > 1$; the cumulative distribution function of X is then $\Pr[X \le x] = 1 - x^{-\alpha}$.

Suppose we discard the β fraction of the mass in the tail of of X, i.e., let x_0 be such that

$$\int_{x_0}^{\infty} \alpha x^{-\alpha} dx = \beta.$$

Since $\beta = \Pr[X > x_0] = x_0^{-\alpha}$ by definition, we get

$$x_0 = \beta^{-1/\alpha}. \qquad (5)$$

Let Y be the random variable truncated at x_0 and rescaled by $1/(1-\beta)$. We first show the following useful inequality.

LEMMA 4. $(1-\beta)(1-\beta^{1-2/\alpha}) \le (1-\beta^{1-1/\alpha})^2$.

PROOF.

$$\begin{aligned}(1-\beta)(1-\beta^{1-2/\alpha}) &= 1-\beta-\beta^{1-2/\alpha}+\beta^{2-2/\alpha}\\ &\overset{\text{am-gm}}{\le} 1-2\sqrt{\beta\cdot\beta^{1-2/\alpha}}+\beta^{2-2/\alpha}\\ &= 1-2\beta^{1-1/\alpha}+\beta^{2-2/\alpha}\\ &= (1-\beta^{1-1/\alpha})^2.\end{aligned}$$

□

LEMMA 5. *If* $\alpha \le 3$, *then* $\mathrm{var}[X] = \infty \gneqq \mathrm{var}[Y]$. *If* $\alpha > 3$, *then* $\mathrm{var}[Y] \le ((1-\beta^{1-1/\alpha})/(1-\beta))^2 \mathrm{var}[X] \le \mathrm{var}[X]$.

PROOF. The easy case is when $\alpha \in (2,3]$, in which case $\mathrm{var}[X] = \infty \gneqq \mathrm{var}[Y]$. For the rest, we will assume $\alpha > 3$.

$E[X] = \alpha/(\alpha-1)$ and $E[X^2] = \alpha/(\alpha-2)$ and so,

$$\mathrm{var}[X] = \frac{\alpha}{(\alpha-2)(\alpha-1)^2}. \tag{6}$$

By integrating the pdf of Y from 0 to x_0 and using the value of x_0 from (5), we obtain

$$E[Y] = \frac{\alpha}{(1-\beta)(\alpha-1)}(\beta^{1-1/\alpha}-1),$$

and

$$E[Y^2] = \frac{\alpha}{(1-\beta)(\alpha-2)}(\beta^{1-2/\alpha}-1),$$

from which,

$$\begin{aligned}\mathrm{var}[Y] \quad = \quad & \frac{\alpha}{(1-\beta)(\alpha-2)}(\beta^{1-2/\alpha}-1)\\ & -\frac{\alpha^2}{(1-\beta)^2(\alpha-1)^2}(\beta^{1-1/\alpha}-1)^2. \qquad (7)\end{aligned}$$

Finally, using (6) and (7),

$$\begin{aligned}&\frac{\mathrm{var}[Y]}{\mathrm{var}[X]}\\ &= \frac{(\alpha-2)(\alpha-1)^2}{(1-\beta)}\left(\frac{1-\beta^{1-2/\alpha}}{\alpha-2} - \frac{\alpha(1-\beta^{1-1/\alpha})^2}{(1-\beta)(\alpha-1)^2}\right)\\ &= \frac{(\alpha-1)^2(1-\beta)(1-\beta^{1-2/\alpha}) - \alpha(\alpha-2)(1-\beta^{1-1/\alpha})^2}{(1-\beta)^2}\\ &\overset{\text{Lemma 4}}{\le} \frac{(1-\beta^{1-1/\alpha})^2}{(1-\beta)^2}\cdot((\alpha-1)^2-\alpha(\alpha-2))\\ &= \left(\frac{1-\beta^{1-1/\alpha}}{1-\beta}\right)^2.\end{aligned}$$

Notice that since $0<\beta<1$ and $\alpha>0$, we have $0<\beta\le\beta^{1-1/\alpha}<1$. □

4. APPLICATIONS OF THE BASIC ESTIMATOR

Recall that our goal is to devise a method to estimate the corpus size, i.e., the number of documents in a search engine corpus. In this section we present two algorithms that achieve this goal. The two algorithms are based on different assumptions about the capabilities of the index. In the first algorithm, we assume that a uniform random document can be obtained from the corpus. In the second algorithm, we do away with the uniform document sampleability assumption, but instead use a different assumption, which will be evident from the description below.

4.1 Corpus size via random documents

Suppose we can obtain a uniform random sample of documents in D. Then, using such a sample, we can estimate the fraction of documents in D that are also in D_A. Then, using D_A and Lemma 1, we can estimate the corpus size.

COROLLARY 6. *If* $p_A = |D_A|/|D|$, *then*

$$|D| = \frac{|A|}{p_A}\cdot W_{A,D}.$$

Thus, by estimating p_A via the random sample of documents, we can estimate the corpus size.

4.2 Corpus size via uncorrelated query pools

Second, suppose uniform random sampling of documents in the corpus is not possible. We show that even under this constraint, the corpus size can be estimated provided we make assumptions about the query pool. The core idea is to use an additional query pool B with the property that B is reasonably uncorrelated with respect to A in terms of occurrence in the corpus (Corollary 8). In other words, we use the independence of the query pools A and B.

Let A, B be two query pools. Let $D_{AB} \subseteq D$ be the set of documents that contain at least one term in A and one term in B. Then, from Lemma 1, we have

COROLLARY 7.

$$\frac{|D_{AB}|}{|A|} = W_{AB,D} = E_{a\in A}\left[\sum_{\substack{d\in D_{AB}\\ d\ni a}} w_d^A\right]$$

Here, notice that the summation is over all documents d that contain a and at least one term in B, i.e., the documents are "filtered" by the query pool B. Thus, Corollary 7 can be used to estimate $|D_{AB}|$.

The significance of Corollary 7 is that it lets us estimate $|D|$ without using any random access to documents in D, modulo appropriate assumptions on A and B. Suppose A and B are uncorrelated, i.e., $\Pr[d \in D_A \mid d \in D_B] = \Pr[d \in D_A] = p_A$, then it is easy to see

COROLLARY 8. *If* A *and* B *are uncorrelated set of terms, then*

$$|D| = \frac{|D_A|\cdot|D_B|}{|D_{AB}|}.$$

Thus, Corollary 8 can be used to estimate the corpus size without resorting to sampling documents in the corpus uniformly at random.

While Corollary 8 is very attractive if the query pool A and B are perfectly uncorrelated, in practice it may be hard to construct or obtain such sets. However, we observe even if the set of terms A and B are correlated, the measure of correlation directly translates to the quality of approximation of the corpus size. More precisely, let $p_{A|B}$ denote $\Pr_{d\in D}[d \in D_A \mid d \in D_B]$. If $c_1 p_A \le p_{A|B} \le c_2 p_A$ for some non-zero constants $c_1 \le c_2$, then it follows along the lines of Corollary 8 that

$$c_1\frac{|D_A|\cdot|D_B|}{|D_{AB}|} \le |D| \le c_2\frac{|D_A|\cdot|D_B|}{|D_{AB}|}.$$

4.3 Size of subsets of the corpus

In a straightforward way, our algorithm in Section 4.1 can be modified to estimate the size of various interesting subsets of the corpus. Subsets may be, for instance, the set of all Chinese documents, the set of documents that have no hyperlinks, or the set of documents that are at least 10K in size. Also note that even more generally, our basic estimator can be used to estimate any weighted sum of the documents, where every document has a weight that can be computed by looking at the document.

4.4 Random sample of subsets of the corpus

Given a subset of the corpus defined by a query pool, our basic estimator in Section 3 can be adapted to produce a uniform random sample from this subset of the corpus. This is done by applying a version of rejection sampling that takes into account the weight of a document. Thus, this method can be used to generate a uniform random document that, for instance, contains at least one US zipcode. Being able to generate a uniform random sample of a subset of the corpus is a powerful primitive and has numerous applications. We leave the details to the full version of the paper.

5. EXPERIMENTS ON TREC

In this section we present our experiments on the TREC collection.

5.1 Data and methodology

The document set D consists of 1,246,390 HTML files from the TREC `.gov` test collection; this crawl is from early 2002 (`ir.dcs.gla.ac.uk/test_collections/govinfo.html`). Our methodology is to first define a query pool, pick sufficiently many sample terms from this query pool, query the index for these sample terms, and then compute the weight of the sample query terms according to (3). All our sampling is done with replacement. For all the experiments, we compute 11 averaged estimators as discussed in Section 3.2 and the final estimate is the median of these 11 averaged estimators. We preprocess the entire data by applying `lynx` on each of the files to process the HTML page and output a detagged textual version. This is so that the meta-data information in the HTML files is not used in the indexing phase. Moreover, this also serves as a data cleaning phase. We tokenize the documents using whitespace as the separator.

An important point to keep in mind is a consistent interpretation of a term "occurring" in a document. This plays a role in two different cases. First, in the answers returned by the index — the index should return all and only documents in which the given query term occurs. Second, in the computation of weight of a term in (1) — we need to know how many terms from A occur in the document. The first of these cases can be handled easily by having a very "strict" definition of "occurring" and checking to see if each document returned by the index for a given query term actually contains the term according to the definition of "occurring". The second case is trickier, unless we have a reasonable understanding of the way the indexer operates. For sake of this experiment, we adopt the safe approach by hand-constructing a straight-forward indexer.

5.2 Corpus size via random documents

We illustrate the performance of our methods using two different query pools and random documents from the corpus. For both the query pools A and B, we estimate p_A and p_B by examining random TREC documents. The experiment in this case has been constructed to remove any systematic bias from the estimate of p_A and p_B in order to evaluate how well the techniques perform when the random samples are good.

5.2.1 *A = set of five-digit numbers*

We choose the first query pool A to be the set of all five-digit numbers, including numbers with leading zeros. Thus, $|A| = 10^5$. Our notion of a term occurring in a document is governed by the regular expression `/^\d{5}$/`. This will, for instance, ensure that $12,345$ or 12345.67 are not valid matches for the term $12345 \in A$. Under these definitions, we have $|D_A| = 234,014$ and so $p_A = 0.1877$ and 94,918 terms in $|A|$ actually occur in some document in D.

The following table shows the error of results of our experiments with set A. Here, error is measured relative to $|D|$, which we know ($|D| = 1,246,390$), as a function of the number of samples used for each averaged estimator.

Samples	100	500	1000	2000	5000
Error (%)	48.50	37.56	13.31	16.12	6.44

As we see, the error of the method is quite low.

Variance reduction. We next investigate whether the performance of this method can be improved by applying the variance reduction techniques presented in Section 3.3. To understand this further, we compute the weights of all terms in A to see if it indeed has a heavy tail. Figure 1 shows the distribution of weights of terms in the set $|A|$. From the figure, it is easy to see that the distribution conforms to a power law (the exponent of the pdf is $\sim$ -1.05). So, there

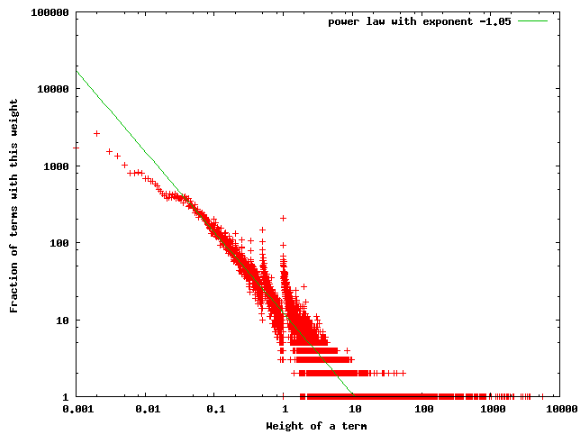

Figure 1: Distribution of weights of terms in $|A|$.

is hope of improving the performance of the method by the variance reduction method outlined in Section 3.3. To identify the candidate elements in A' — the terms with highest weights — we resort to sampling once again. We randomly sample documents from the corpus, and for each term $a \in A$ that occur in a document d, we maintain a histogram of its weight according to w_d^A as in (1); note that these weights are in fact approximations to their actual weights w_a^A as in (2). We finally sort the terms in decreasing order of their accumulated weights and declare the terms that contribute to the top 75% of the weights to be present in A'.

This operation defines a new query pool $A \backslash A'$, upon which our methods apply as before. Thus, the correctness of this approach is not dependent on the exact determination of the most frequent terms, or even upon uniform sampling. The method is always correct, but the variance will only benefit from a more accurate determination. The results are shown below.

Samples	100	500	1000	2000	5000
Error (%)	14.39	16.77	19.75	11.68	0.39

We can see that overall this method obtains estimates with significantly lower error, even with few samples.

5.2.2 B = *set of medium frequency words*

We repeat the same experiments with a different set B of terms. This time we want to choose B with two properties: none of the terms in B matches too many documents and B is reasonably large. The former property will reduce the variance of the sampling steps if the occurrence of terms is correlated positively with the weights. We provide some evidence towards this. See Figure 2. We first extract all terms

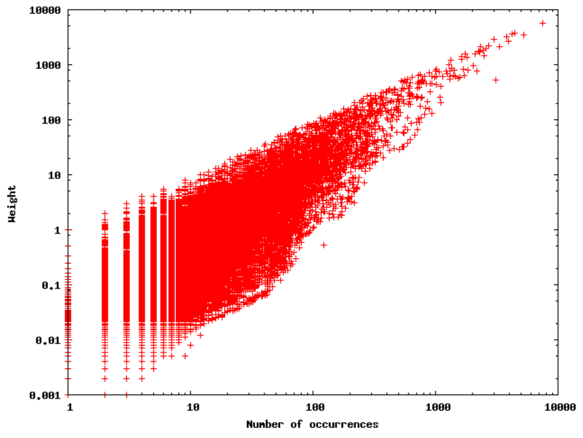

Figure 2: Correlation of weights and number of occurrences of terms in A.

in the document set D using whitespace separated tokenization and then we sort the terms according to their frequency of occurrence. We then pick B to be the terms (that are not purely numbers) that occur from position 100,000 to position 200,000 in this list. Thus, $|B| = 100{,}000$. Under these definitions, we obtain $|D_B| = 396,423$ and so $p_B = 0.3180$.

Samples	100	500	1000	2000	5000
Error (%)	3.51	3.66	1.24	0.95	1.80

We see that the method performs extremely well and this can be attributed to our careful choice of the set B. In fact, our experiments showed no appreciable improvement when we applied the variance reduction method to the set B. This is to be expected, as B is specifically constructed so that no term occurs significantly more often than any other term, and so no term will introduce substantial skew into the measurement.

5.3 Corpus size via uncorrelated query pools

We need to construct query pools A and B that are reasonably uncorrelated. To do this, we first try A and B as defined before. Since we have the corpus available, we can actually measure the amount of dependence between the sets A and B. We explicitly calculate $p_{A|B}$ and p_A. The values are $p_A = 0.1877$ whereas $p_{A|B} = 0.2628$ indicating some correlation between the term sets. To reduce the correlation, we modify the sets D_A and D_B to be slightly different.

We set D'_A to be the set of documents that contain *exactly one* term from A; D'_B is defined analogously. We do this in order to reduce the potential correlation caused by large documents. Using this, we calculate $p'_A = 0.1219, p'_B = 0.1437$, and $p'_{A|B} = 0.1455$, indicating a significant reduction in correlation.

Modifying Lemma 1, we can estimate $|D'_A|$ and $|D'_B|$. We use Corollary 7 to estimate $|D_{AB}|$. We proceed as before, except that in computing the weight of a sample term from B, we discard documents that do not contain any term from A. The following table shows the error of the method.

Samples	1000	2000	5000
Error (%)	27.64	23.01	21.32

As we see from the results, the method obtain a reasonable estimate of the corpus size, without using any random sample of documents. Obviously, the accuracy of this method can be improved if we work with even less correlated query pools.

6. EXPERIMENTS ON THE WEB

In this section we present our experiments on the Web. We used the public interface of three of the most prominent search engines, which we refer to as SE$_1$, SE$_2$, and SE$_3$. We present three series of results for the Web. First we use the basic estimator defined in Section 3.1 to compute the relative sizes of the three engines. We then use the random document approach and the uncorrelated query pool approach defined in Section 4 to compute absolute sizes of the "visible" portion of the engines. (See below.)

6.1 Method

Our methodology for the Web is similar to the one we used for TREC. We first define a query pool, sample sufficiently many terms from the query pool, and compute the weights as defined in (3). We postprocess the results using `lynx` to remove the HTML markup of on each of result pages. Then we compute the weights based on this cleaned version of the result pages, using whitespace tokenization. We use the same definition of a term "occurring" in a document as in the TREC experiment — the cleaned version of the document must contain the queried term. This eliminates documents in which the query term does not appear in the document, including documents in which the query term appears only in anchor text, dead pages (404), and pages that match stemmed versions of the query term. Furthermore, since the capabilities of `lynx` as an HTML parser are limited, many pages are "invisible" to our method, in particular most pages containing Java script, frames, Flash, etc. Thus the absolute sizes that we are reporting, are only estimates for the sets "visible" to our methodology. Assuming that each search engine carries the same proportion of "visible" pages, we can obtain relative sizes of the search engines, but absolute estimates are still an elusive goal.

6.2 Results

We first compute the relative sizes of engines SE_1, SE_2, and SE_3, using the our basic estimator. We define A to be the set of all eight-digit numbers, including numbers with leading zeros, thus, $|A| = 10^8$, and follow the approach detailed for the TREC experiments. The following table shows the results of our experiments with query pool A.

Engine	SE_1	SE_2	SE_3
Samples	3486	3529	3433
$W_{A,D}$	0.29	0.51	0.08

If we assume that p_A is the same for all the three search engines, i.e., that the three engines index the same proportion of pages with eight-digit numbers, the above values provide the relative sizes of the engines corpus. However, p_A varies from engine to engine. We used a random sample of pages provided to us by Bar-Yossef and Gurevich produced according to the random walk approach described in their work [4]. The following table shows p_A for the three engines and the sample sizes.

Engine	SE_1	SE_2	SE_3
Samples	199	137	342
p_A	0.020	0.051	0.008

Using these values, we can easily compute the absolute sizes of the "visible" portion of the three search engines (in billions) as below.

Engine	SE_1	SE_2	SE_3
Size	1.5	1.0	0.95

Next, we used the uncorrelated query pool approach to estimate the absolute corpus sizes. The query pool A is again the set of all eight-digit numbers, and as for TREC, we chose the query pool B to be the medium frequency words, which was determined from a histogram of term counts from the index of one of the search engines. We assume that these words are also the medium frequency words on the other two engines. Furthermore, we also verified that when queried, all three engines always returned less than 1000 results for all our samples from pool B. However, for the Web there is no straightforward way to verify that pool A and pool B are indeed independent or to estimate their correlation.

The following table shows the resulting estimates for the "visible" portion of the three search engines (in billions) using uncorrelated query pools.

Engine	SE_1	SE_2	SE_3
Size	2.8	1.9	1.1

As can be readily seen, the estimates are now larger although still much less than published values for the entire corpus of these engines while the relative sizes are fairly consistent. The next section explains why this happens.

6.3 Discussion and caveats

Even though our methods are intended to produce absolute estimates for the corpus size, they are still affected by many idiosyncracies that exist in web documents. Here we list some of the caveats while drawing conclusions from the results above.

1. Even though our methods are fairly generic, they end up measuring only a large slice of the corpus (rather than the corpus in its entirety), what we call the "visible" corpus. In particular, our methods exclude documents in the so-called frontier. These documents are not indexed in full, nevertheless the search engine is aware of their existence through anchortext and might return these documents as results to queries (especially when these queries match the anchortext). Our method might end up discarding these documents since the query term may not explicitly occur in the document. Hence, our method will underestimate the corpus size. A similar comment to the many types of documents that are not parsed by our HTML parser (`lynx`). Although `lynx` has the advantage of being very consistent, in future work we intend to use a more sophisticated parser.

2. Even though we chose our query pools carefully, for about 3% of the queries in the numbers query pool, the search engines return more than 1000 results. For such queries, we do not have access to the result documents after the first 1000. Thus in this cases we end up underestimating the weight of the query, and consequently, the corpus size.

3. Even though the search engine may return fewer than 1000 results for a given query, to promote diversity, it might restrict the number of results from a single host/domain. For the engines we tested this limit is set to two results per host/domain.

4. The number of samples used to estimate p_A is fairly small, since these samples were graciously provided by the authors of [4] and their generation method is fairly laborious. These samples are uniformly random *over a large portion of the underlying corpus restricted to English documents*, namely those pages that match at least one query from the pool used in [4]. Thus what we estimate using these samples, is the number of "visible" pages that match at least one query from the pool used in [4], which is restricted to English only. This explain why these estimates are substantially lower than the second set of estimates, while the relative sizes are fairly consistent.

5. In the uncorrelated query pool approach, our choice of the query pool B has a natural bias against non-English documents. In other words, the pools A and B are presumably uncorrelated only with respect to the English portion of the corpus. Once again, this will result an underestimate of the actual corpus size. For instance the addition of one billion pages in Chinese that do not contain contiguous eight digit strings would be invisible to our approach.

7. CONCLUSIONS

We addressed the problem of estimating the size of a corpus using only black-box access. We constructed a basic estimator that can estimate the number of documents in the corpus that contain at least one term from a given query pool. Using this estimator, we compute the corpus size using two different algorithms, depending on whether or not it is possible to obtain random documents from the corpus.

While the ability to randomly sample the corpus makes the problem easier, we show that by using two query pools that are reasonably uncorrelated, it is possible to obviate the need for random document samples. En route, we also obtain a novel way to provably reduce the variance of a random variable where the distribution of the random variable is monotonically decreasing; this technique may be of independent interest.

We applied our algorithms on the TREC collection to measure their performance. The algorithms that uses random document samples perform quite well as expected. More surprisingly, by carefully constructing query pools that are reasonably uncorrelated, we show that it possible to estimate the corpus size to modest accuracies. We also apply our algorithms to estimate the "visible" corpus size of major web search engines.

Acknowledgments

We thank Ziv Bar-Yossef and Maxim Gurevich for their invaluable help. We thank Anna Patterson for helpful comments, suggestions, and criticisms.

8. REFERENCES

[1] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. In *Proc. 14th International Conference on World Wide Web*, pages 245–256, 2005.

[2] J. Bar-Ilan. Size of the web, search engine coverage and overlap – methodological issues. Unpublished, 2006.

[3] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proc. 26th International Conference on Very Large Data Bases*, pages 535–544, 2000.

[4] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. *Proc. 15th International World Wide Web Conference*, pages 367–376, 2006.

[5] K. Bharat and A. Broder. Mirror and mirror and on the web: A study of host pairs with replicated content. *Computer Networks*, 31(11-16):1579–1590, 1999.

[6] K. Bharat and A. Z. Broder. A technique for measuring the relative size and overlap of public web search engines. *Computer Networks*, 30(1-7):379–388, 1998.

[7] K. Bharat, A. Z. Broder, J. Dean, and M. R. Henzinger. A comparison of techniques to find mirrored hosts on the WWW. *Journal of the American Society for Information Science*, 51(12):1114–1122, 2000.

[8] A. Z. Broder. Web measurements via random queries. Presentation at the Workshop on Web Measurement, Metrics, and Mathematical Models (WWW10 Conference), 2000.

[9] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[10] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proc. 14th International Conference on World Wide Web (Special interest tracks & posters)*, pages 902–903, 2005.

[11] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. *Computer Networks*, 33(1-6):295–308, 2000.

[12] S. Lawrence and C. Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[13] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[14] K.-L. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In *Proc. 11th International Conference on Information and Knowledge Management*, pages 652–654, 2002.

[15] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the world wide web. In *AAAI Fall Symposium on Using Uncertainty Within Computation*, pages 121–128, 2001.

[16] C. J. van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, 1979.

[17] S. Wu, F. Gibb, and F. Crestani. Experiments with document archive size detection. In *Proc. 25th European Conference on IR Research*, volume 2633 of *Lecture Notes in Computer Science*, pages 294–304. Springer, 2003.