View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Scholarship at UWindsor

University of Windsor
Scholarship at UWindsor

Computer Science Publications | Department of Computer Science

2010

Ranking Bias in Deep Web Size Estimation Using Capture Recapture Method

Jianguo Lu
University of Windsor

Follow this and additional works at: http://scholar.uwindsor.ca/computersciencepub

 Part of the Computer Sciences Commons

Recommended Citation

Lu, Jianguo. (2010). Ranking Bias in Deep Web Size Estimation Using Capture Recapture Method. *Data and Knowledge Engineering*, 69 (8), 866-879.
http://scholar.uwindsor.ca/computersciencepub/2

This Article is brought to you for free and open access by the Department of Computer Science at Scholarship at UWindsor. It has been accepted for inclusion in Computer Science Publications by an authorized administrator of Scholarship at UWindsor. For more information, please contact scholarship@uwindsor.ca.

Ranking Bias in Deep Web Size Estimation Using Capture Recapture Method

Jianguo Lu

Preprint submitted to Elsevier *March 12, 2010*

Ranking Bias in Deep Web Size Estimation Using Capture Recapture Method

Jianguo Lu

School of Computer Science, University of Windsor
401 Sunset Avenue, Windsor, Ontario, Canada.
email: jlu@uwindsor.ca

Abstract

Many deep web data sources are ranked data sources, i.e., they rank the matched documents and return at most the top k number of results even though there are more than k documents matching the query. While estimating the size of such ranked deep web data source, it is well known that there is a ranking bias– the traditional methods tend to underestimate the size when queries overflow (match more documents than the return limit). Numerous estimation methods have been proposed to overcome the ranking bias, such as by avoiding overflowing queries during the sampling process, or by adjusting the initial estimation using a fixed function.

We observe that the overflow rate has a direct impact on the accuracy of the estimation. Under certain conditions, the actual size is close to the estimation obtained by unranked model multiplied by the overflow rate. Based on this result, this paper proposes a method that allows overflowing queries in the sampling process.

Keywords: Deep web, ranked data source, estimators, capture-recapture.

1. Introduction

The deep web [7, 27] is the content that is hidden behind HTML forms, web service interfaces, or other types of programmable web APIs. Probing the size of a deep web data source has become an important problem ever since the web emerges. With a wide availability of web services and programmable web interfaces, the size estimation becomes even more important and less costly. Nowadays almost all organizations or communities have web presence, and most of them provide a keyword based search interface. The size of these searchable data sources can reflect the size of the organizations, or the intensity of the activities in these organizations. For example, by estimating the size of the data sources, we can learn the number of products being sold by an e-commerce company, the number of books kept in a university library, the number of articles published by a newspaper, or the number of blogs posted in a social networking

web site. Such information is of interest to general public, sometimes vital for attracting advertisements from which many web sites live upon.

Estimating the size of a hidden data source is also an essential sub problem in data source sampling and selection [12, 13, 21, 36, 40], and deep web crawling [27, 26, 29, 22, 31, 33]. In deep web crawling, we need to know whether most of the data have been harvested. Without the knowledge of the data source size, it is difficult to decide when to stop the crawling process, and how to evaluate the performance of the data extractors.

There have been tremendous research on data source size estimation [4, 5, 8, 9, 11, 13, 34, 36, 40, 42], all are more or less based on the traditional capture-recapture method [1, 14, 16] that was first developed in ecology for the estimation of wild animals. The basic idea is to capture a collection of animals as randomly as possible, mark the captured animals and release them. Then capture another sample and count the duplicates with the previous captures. With this data various approaches have been proposed to estimate the animal population size. The method also find its applications in computer science, including software defect estimation [32], phishing population detection [39], web directory size evolution [3], and in particular, data collection size estimation [34] [25] [11].

When individuals in a population, or documents in a data source, have equal probability of being captured, there are various mature estimation methods [17]. However, when the capture probabilities vary, it is well known that the traditional methods tend to underestimate a data collection size [1].

Unequal capture probability can be caused by various reasons. In ecology, animals may have varying capture probability due to their age etc. In data source size estimation, documents can have unequal capture probabilities for reasons such as query bias and ranking bias that are first introduced by Bharat et al. [8].

Query bias is caused by the unequal probability of a document being *matched* with a query. A large document may have a higher probability of being matched by a query because it contains more words or queries.

Ranking bias is caused by the probability that a document can be *returned*. Note that in many data sources not every *matched* document is *returned* to a user. For example, Google API returns only the top 1000 matched documents, while Yahoo BOSS API returns only the top 100 documents. One salient feature of deep web data sources is that many of them are ranked data sources, i.e., the data sources sort the documents according to a criteria, and return only at most top k documents for each query, even though there are more than k matches. Queries that match more than k documents are called $overflowing$ queries.

There are at least two approaches to coping with the query bias and ranking bias. One is to control the sampling process so that each document has an equal probability of being matched and returned [4, 5, 6, 37]. For example, to overcome query bias, a matched document can be rejected according to a probability associated with document size. To overcome ranking bias, overflowing queries are omitted so that all the matched documents are returned. When a random sample is obtained, a simple estimator can be applied to such random sample

data. We call this the *sampling based approach*.

The other approach does not strive to obtain random samples, as it may be too costly or sometimes impossible. Instead, people use the data as they are, and develop estimators to compensate the bias. We call this the *estimator based approach*, which is widely studied in Ecology. In the area of wild animal population estimation, people developed various (multiple) capture-recapture estimation methods to deal with unequal catch probabilities of animals. There are several well known methods such as Chao et al's coverage method [14], and Otis' Jackknife method [1].

This paper takes the estimator based approach. Our previous paper [25] studies the query bias and assume that the density of the overflowing queries are not high. That assumption restricts the estimator being applied to either relatively small data sources or large data source with no return limit. This paper focuses on the ranking bias so that our estimator can be applied to ranked large data sources. In this study we observe that the overflow rate, the ratio between the number of matches and the return limit, has a direct impact on the estimation result. Under certain conditions, the actual size is close to the estimation obtained by unranked model multiplied by the overflow rate. Based on this result, this paper proposes a method that allows overflowing queries in the sampling process.

In the realm of estimator based approach for data collection size estimation, several other estimation methods have also been proposed to overcome various biases [34, 42]. However, they typically try to compensate the biases indistinctively. None of them dissects the causes of various biases and focuses on ranking bias only. For example, Shokouhi et al [34] correct the bias by establishing a fixed relationship between the initial estimation and the corrected one. In their estimation method, factors such as the return limit of a data source is not considered.

2. Capture recapture method

2.1. Basic concepts of capture recapture method

Capture-recapture method was originally developed in ecology and used to estimate the size of an animal population [1] [30]. In the estimation process, animals are captured, marked, and released in several trapping occasions. The data collected during the process, including the number of capture occasions, the recaptured animals, and the distinct animals captured, allow one to estimate the total population. This estimation process corresponds nicely to the query based estimation of data collection sizes, where animals correspond to documents, and a trapping occasion corresponds to the process of sending a query and retrieving a set of matched documents from a data source.

We summarize the statistics that are used in our estimation methods as below:

- N: the actual number of documents in a data source;

- t: the number of queries that are sent to a data source;
- m_j: the number of matched documents for query j. $1 \leq j \leq t$. $n = \sum_{j=1}^{t} m_j$ is the sample size, i.e., the total number of matched documents in the estimation process;
- u_j: the number of new documents retrieved by query j. $1 \leq j \leq t$. $M_i = \sum_{j=1}^{i-1} u_j$ is the total number of marked or unique documents that are retrieved before query i. Note that $M_1 = 0$, and $M_2 = m_1$. Let $M = M_{t+1}$ denote the total number of distinct documents that are retrieved by all the queries in the estimation process;
- d_i: the number of duplicate documents retrieved by query i. $d_i + u_i = m_i$;
- k: the maximal number of returns from a ranked data source, even if there are $m_j > k$ number of matches.
- $OR = n/M$: the Overlapping Rate up to the t-th query, i.e., the ratio between the sample size and the distinct documents;
- $P = M/N$: the percentage of the documents that has been sampled, i.e., the ratio between the distinct documents and the actual size.

If all the documents have an equal probability of being matched by a query, and all the matched documents are returned, we have the simplest model for which many estimators have been developed. The classic estimator is the famous Petersen estimator [30] that can be applied only to two capture occasions:

$$\hat{N}_{Petersen} = \frac{m_2 M_2}{d_2}. \tag{1}$$

This estimator can be derived using maximally likelihood method. The problem of this estimator is that d_2 could be zero when m_2 and M_2 are not large enough. According to the birthday paradox, in general m_2, M_2 should be greater than $\sqrt{N}$ in order to have overlaps between the results of two queries. Unfortunately, many queries do not have that many matches.

One approach to solving the problem is by obtaining two large samples, each are produced by many queries instead of just one query [11].

Another approach is expanding the estimator to multiple capture occasions or queries, and taking the weighted average of the estimations. i.e.,

$$\hat{N} = \frac{\sum_{i=2}^{t} w_i m_i M_i / d_i}{\sum_{i=2}^{t} w_i}$$

When weight $w_i = d_i$, it is the classical Schnabel estimator [30] for multiple captures:

$$\hat{N}_{Schnabel} = \frac{\sum_{i=2}^{t} m_i M_i}{\sum_{i=2}^{t} d_i}. \tag{2}$$

When weight $w_i = d_i M_i$, it is the Schumacher estimator [35]:

$$\hat{N}_s = \frac{\sum_{i=2}^{t} m_i M_i^2.}{\sum_{i=2}^{t} d_i M_i} \tag{3}$$

Unlike two capture occasions, the MLE (Maximum Likelihood Estimator) for multiple capture model does not have a closed form solution. Hence both Schnabel and Schumacher estimators are approximate estimators. However, they are widely accepted as good estimators with very small bias and variance when animals (or documents) are captured with equal probability.

In reality, individuals, be it documents or animals, seldom have equal capture probability. For animals, young animals may be easier to be captured because they are more active. For documents, large documents are easier to be retrieved by a query because there are more words in those documents.

For this kind of heterogeneous population where each individual has a unequal catachability, the estimation is notoriously difficult [1]. MLE technique can no longer be used to derive an estimator because there can be as many as N+1 parameters: N and and capture probabilities $p_1, p_2, \ldots, p_N$. Estimating this many parameters from the capture data is not possible. Although there are several empirical estimators proposed for this model, including the Jacknife estimator [30] and Chao [14] method, both can be only applied to small population with hundreds of individuals, and require large sample size.

2.2. Application of capture recapture methods in data source size estimation

Liu et al. [23] first applied the capture recapture method in the estimation of data source size. The estimator they used is the traditional Petersen estimator (Equation 1) that can be applied to two capture occasions only.

Shokouhi et al [34] are the first who proposed to use *multiple* capture recapture method, or Capture with History (hereafter CH) method, to estimate a data source size. More recently, Thomas proposed another estimator based on multiple capture recapture methods [38]. Shokouhi et al. introduced the traditional estimators in ecology, such as the famous Schumacher and Eschmeyer estimator [35] as shown in Equation 3. Since Schumacher estimator works only when each document or animal has an equal probability of being captured, Shokouhi noticed that $\hat{N}_s$, the estimate obtained by Schumacher method, is consistently smaller than the actual value N. Furthermore, they observe that there is a fixed relationship between $\hat{N}_s$ and N, and used regression to establish such a relation.

Xu et al.'s work [42] is the first to apply maximum likelihood (ML) method in data source size estimation in the context of multiple capture recapture method. When each document may have a different capture probability, it is impossible to estimate all those probabilities using ML method because there are more parameters than data available. Hence it is necessary to reduce the number of parameters by assuming that the capture probabilities follow a distribution. Xu et al. model the capture probability using a logistic function that relies on

Table 1: Models, assumptions, and their estimators

Model name	Assumptions: each document has equal probability of being matched	Assumptions: all matched documents are returned	Estimator
M_0	yes	yes	$\hat{N}_0 = M/(1 - OR^{-2.1})$
M_{0r}	yes	no	$\hat{N}_{0r} = \mathcal{OF} \times M/(1 - OR^{-2.1})$
M_h	no	yes	$\hat{N}_h = M/(1 - OR^{-1.1})$
M_{hr}	no	no	$\hat{N}_{hr} = \mathcal{OF} \times M/(1 - OR^{-1.1})$

factors including the length of the document, its static ranking, and the term frequency of the query in the document.

Broder et al. [11] combined the sampling based and estimator based methods, and used Petersen estimator (Equation 1) in a very creative way in order to overcome both the query bias and rank bias. Here the matched documents are not obtained by one single query. Instead, the first (M_2) and the second (m_2) captures are obtained by query pools that can be very large, possibly including tens of thousands of queries. For example, a query pool can be all the eight digit numbers, or all the medium frequency terms. Since the first and second captures are not obtained by single queries, and those two captures are largely independent, there is no query bias. By using medium frequency words, the possibility of having overflowing queries is low, therefore avoided the ranking bias. Since both m_2 and M_2 are rather large, the method can handle very large data source without using multiple capture recapture method. Because the query pool is rather large, it is impractical to fire all the queries to decide the number of documents that can be retrieved using the query pool. Hence, from the query pool a set of sample queries are randomly selected and sent to the data source. All the matched documents are downloaded and analyzed to obtain the weight [11] of each query. Based on the average weight of the queries from the sample, the number of the documents that can be obtained by the query pool can be estimated by multiplying the average weight by the query pool size.

3. Our approach

3.1. Models

Different types of data sources require different estimators. In order to categorize various types of data sources, we classify them into four models, based on whether two assumptions are imposed on the data source, i.e., whether random documents can be obtained, and whether all the matched documents are returned. The models, assumptions, and their corresponding estimators are tabulated in Table 1.

While models M_0 and M_h are common in capture-recapture studies [1], models M_{0r} and M_{hr} are unique in our study. This is because in ecology studies

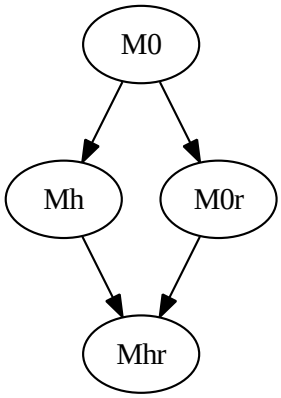

Figure 1: Model hierarchy

usually all the obtained data are utilized. On the other hand, in deep web it is common to return only top-k documents.

If all the documents have an equal probability of being matched by a query, and all the matched documents are returned, we have the simplest model M_0. The subscript 0 denotes zero variation of the match probability. Many estimators have been developed for this simple model. For example, paper [25] derived the relationship between P and OR as

$$P = 1 - OR^{-2.1}. \tag{4}$$

Hence the estimator for M_0 is

$$\hat{N}_0 = M/P = M/(1 - OR^{-2.1}), \tag{5}$$

Paper [25] showed that this estimator has a smaller bias and variance than the classical Schumacher estimator (Equation 3).

Model M_{0r} is a simplified version of the ranked model. The assumptions are 1) the documents have an equal probability of being matched, and 2)the matched documents are sorted according to a static ranking and only the top k documents are returned. An estimator for this model is derived in Section 4.

M_h is the model for unranked data sources where the assumptions are 1) the documents are heterogeneous, i.e., they have unequal probabilities of being matched; and 2) all the matched documents are returned. [25] gives an estimator for this kind of data sources. Its simplified version is

$$\hat{N}_h = M/(1 - OR^{-1.1}) \tag{6}$$

Model M_{hr} is the heterogeneous and ranked model that will be discussed in Section 5.

The relationships between the models can be depicted in Figure 1. Model M_0 has very strong assumptions, hence it is hard to find direct applications. To

use this model directly, random documents have to be obtained, which is a very costly process [6]. Besides, in this model no overflowing queries are allowed. Despite the limited application of this model, it is a good starting point to understand the estimation problem.

For Models M_{0h} and M_h, each relaxes one assumption, hence makes it harder to estimate. On the other hand, it moves closer to real applications. Model $_{hr}$ removes both of the assumptions and is the closest to many real data sources.

3.2. Method

Our proposed method is detailed in Algorithm 1. There are two stages, i.e., data collection and estimation. The data collection process of our proposed estimation method is similar to most of the capture-recapture methods, except that in each sampling process the sample documents are obtained by random queries. The estimation method differs from other capture-recapture methods in that we use OR to estimate. In our method documents have unequal probability of being matched, therefore the algorithm gives the estimations for models M_h and M_{hr} only.

Algorithm 1: Outline of estimation algorithm.

Input: *Dictionary*, t, k, *DataSource*
Output: $\hat{N}$, the estimation of the size of *DataSource*
$i = n = M = T = 0$;
while $i < t$ **do**
  Randomly select a single term query from *Dictionary*;
  send the query to *DataSource* ;
  $m =$ number of matched documents ;
  $r = min(k, m)$; //actually returned documents ;
  $u = min(k,$ number of new documents returned$)$;
  $T+ = m$; // total number of matched documents;
  $n+ = r$; // total number of returned documents;
  $M+ = u$; // total number of unique documents;
  $i++$;
end
$OR = n/M$;
$\mathcal{OF} = T/n$;
$\hat{N} = \mathcal{OF} \times M/(1 - OR^{-1.1})$

Please note that k is the return limit imposed by the data source. M denotes the number of distinct documents, n the number of sampled documents, T the total number of matched documents.

This algorithm applies to both models M_h and M_{hr} although only the estimator for M_{hr} is used in the algorithm. In the case of model M_h, the number of matched documents is equal to the number of returned documents, i.e., $m = r, T = n$, hence $\mathcal{OF} = 1$ and the estimator is reduced to the estimator for M_h.

In this algorithm, there are several uncertainties undecided yet. One is the input parameter *Dictionary*. In the experiments reported in the following sections, we used Webster dictionary. Other dictionaries, such as a collection of terms from newsgroups posting, are also used. We find the results obtained by different dictionaries are similar.

The other uncertainty is the number of queries to be sent, i.e., the parameter t. While larger t will always produce better result, our experiments show that 50 to 100 number of queries are good enough. A large data source may not necessarily need a large t, because in this case each query returns more documents than smaller data sources.

Next we will derive the estimator, starting from a simplified model M_{0r}.

4. Ranking bias: Model M_{0r}

4.1. Derivation

A simple model M_{0r} of ranked data source can be described as below:

Given N number of ranked documents labeled $d_1, d_2, \ldots, d_N$, where d_i is ranked higher than d_j if $i < j$, where $1 \leq i, j \leq N$. Suppose that in each time m documents are matched by a query and only the top k documents from the matched documents are returned, where $k \leq m$.

Note that in this section we assume that there is no query bias, i.e., when m documents are selected from a data source, each has the same probability of being matched. That is why we call the model M_{0r} – it is ranked, yet when selecting the m documents each has an equal probability of being matched. Next section we will relax this assumption to allow for heterogeneous match probability.

Let p_i, $1 \leq i \leq N$, denote the probability that the document d_i is captured. When $i \leq k$, document d_i will be returned if it is matched since it will be among the top k documents. Overall, there are $\binom{N}{m}$ number of ways to select m documents from a total population of N, among them there are $\binom{N-1}{m-1}$ combinations that d_i is selected. Hence the probability that d_i is returned is $\binom{N-1}{m-1}/(k\binom{N}{m})$.

When $i > k$, d_i is among the top-k matched documents only when there are at most $k-1$ documents selected from $d_1, \ldots, d_{i-1}$. Thus the combinations when d_i is selected and among the top k elements are $\sum_{j=0}^{k-1} \binom{i-1}{j}\binom{N-i}{m-1-j}$. Hence we have:

$$p_i = \begin{cases} \dfrac{\binom{N-1}{m-1}}{k\binom{N}{m}} & i \leq k \\ \dfrac{\sum_{j=0}^{k-1} \binom{i-1}{j}\binom{N-i}{m-1-j}}{k\binom{N}{m}} & k < i \leq N \end{cases} \tag{7}$$

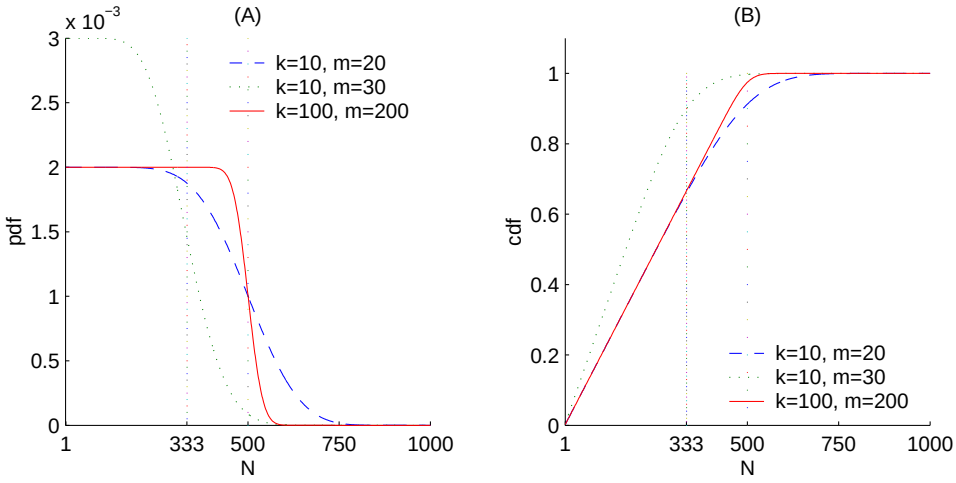

Figure 2: Distribution of p_i. N=1,000.

Note that $\sum_{j=1}^{N} p_i = 1$, and Equation 7 can not be simplified further by Vandermonde's convolution because j does not iterate up to N.

Figure 2 draws the probability density function (PDF) and cumulative probability function (CDF) of p_i for several combinations of m and k, where $N = 1000$, and $1 \leq i \leq N$. Figure 2 (A) shows that p_i drops to near zero when $i > (k/m)N$. In particular, if m/k has the same value, p_i drops at a faster rate when k becomes larger.

For example, in both the combinations $(k = 10, m = 20)$ and $(k = 100, m = 200)$, p_i drops to near 0 when $i = 500 = N(k/m)$. However, the solid line, the combination $(k = 100, m = 200)$, drops almost vertically to near 0. Sub figure (B), which is the PDF of p_i, shows that when i is around $(k/m)N$, the cumulative probability is close to one.

Roughly speaking, Figure 2 says that a ranked data source only exposes the top $(k/m)N$ number of documents, especially when k is large. The remaining $(1 - k/m)N$ documents are highly improbable of being sampled.

The solid line in Figure 2 (A) prompts us that model M_{0r} corresponds to a simpler model M_s where the top $(k/m)N$ documents have the probability $m/(kN)$ of being captured, while the remaining $(1 - k/m)N$ documents have zero probability of being retrieved.

In order to quantify the relationship between M_s and M_{0r}, we resort to the coefficient of variation (CV) of the probabilities p which is defined as

$$\gamma = \frac{1}{\bar{p}} \sqrt{\sum_{i=1}^{N} (p_i - \bar{p})^2 / N} \tag{8}$$

Table 2: γ_r^2 computed by Equation 7 for various combinations of k and m/k obtained by simulation. N=1,000. It shows that the estimation is rather accurate when k is not very small.

	m/k					
k	2	3	4	5	6	7
10	0.75	1.57	2.40	3.23	4.06	4.89
20	0.82	1.70	2.58	3.46	4.35	5.24
40	0.87	1.79	2.71	3.64	4.57	5.51
60	0.90	1.83	2.78	3.72	4.68	5.64
80	0.91	1.86	2.82	3.78	4.75	5.72
100	0.92	1.88	2.84	3.82	4.80	5.80

Note that for the simple model M_s its γ_s^2 is

$$\begin{aligned}\gamma_s^2 &= N\sum_{i=1}^{i=N} p_i^2 - 1 \\ &= N\sum_{i=1}^{i=(k/m)N} p_i^2 - 1 \\ &= N\frac{k}{m}N\left(\frac{m}{kN}\right)^2 - 1 \\ &= \frac{m}{k} - 1 \end{aligned} \tag{9}$$

To compare with the CV in M_s, we calculate the CV of M_{0r} using Equation 7. The result is shown in Table 2. From the table we can see that while CV varies with different values of N, m, k, γ_r^2 is asymptotically equal to $m/k - 1$.

Hence M_{0r} can be reduced to M_s. In this model, any estimation method actually works on a fraction of the data source, which contains the top $(k/m)N$ documents only. Thus the estimator for model M_{0r} should be

$$\begin{aligned}\hat{N}_{0r} &= (m/k)\hat{N}_0 \\ &= (m/k)\frac{M}{1 - OR^{-2.1}}\end{aligned} \tag{10}$$

Equation 10 can be also explained using binomial distribution. N documented are stratified into two layers.The top layer contains the top $(k/m)N$ documents, while the bottom layer contains the remaining $(1 - k/m)N$ documents. When randomly selecting a document from the data source with equal probability in model M_{0r}, the probability of a document belonging to the top layer is k/m, and the probability belonging to the bottom layer is $1 - k/m$. After selecting m times, the number of times the selections hitting the top layer follows a binomial distribution $X \sim B(m, k/m)$. Its mean is $m(k/m) = k$ and the variance is $k(1 - k/m)$.

In another word, when selecting m documents, in average k of them are from the top layer. Around $\sqrt{k(1 - k/m)}$ number of documents can be from outside

Table 3: Fraction of the documents retrieved from bottom layer for some combinations of k and m.

k	m	frac in bottom
10	20	0.223
20	40	0.158
100	200	0.070
1000	2000	0.022

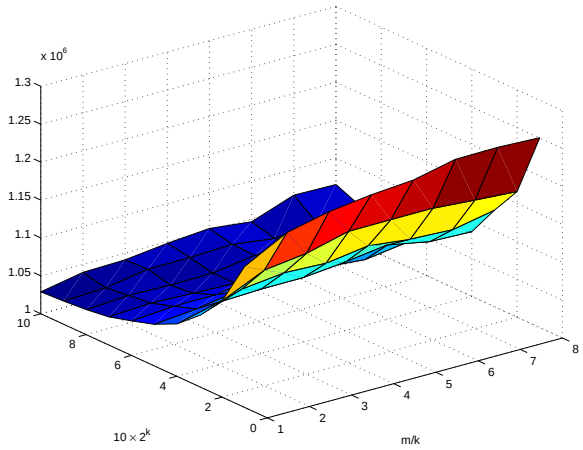

Figure 3: $\hat{N}_{0r}$ for various m and k. $N = 1,000,000$. The estimations are accurate when k is not very small.

the top layer of the data source, i.e., a fraction $\sqrt{k(1-k/m)}/k$ of k documents can be from the bottom layer.

Table 3 tabulates the relative standard deviations $\sqrt{k(1-k/m)}/k$ for some combinations of m and k. It can be seen that when k increases, fewer documents can be retrieved from the bottom layer. This indicates that Equation 10 is a rather accurate estimator when k is not very small, although it is positively biased.

When k is small, we suggest the use of the following adjusted estimator:

$$\hat{N}'_{0r} = (\gamma_r^2 - 1)\hat{N}_0 \tag{11}$$

4.2. Simulation

In the simulations and experiments in this paper, we use relative bias (RB) and relatively standard deviation (RSD) to evaluate the method.

Let N denote the actual size of the collection, $\hat{N}$ the estimation. Suppose that there are i number of estimations. The expectation of $\hat{N}$, denoted as $E(\hat{N})$, represents the mean of i estimations, i.e.,

$$E(\hat{N}) = (\hat{N}_1 + \hat{N}_2 + ... + \hat{N}_i)/i.$$

Table 4: $\hat{N}_{0r}$ by simulation of 30 combinations of k and m/k over 100 runs. RB and RSD are over 100 trials. N=1,000,000.

	m/k											
	2		3		4		5		6		7	
k	RB	SD	RB	SD	RB	SD	RB	SD	RB	SD	RB	SD
10	0.182	2,373	0.215	3,277	0.230	4,071	0.239	4,218	0.244	4,695	0.249	5,499
20	0.130	2,586	0.151	2,775	0.160	4,290	0.166	3,780	0.170	5,161	0.172	4,815
100	0.065	2,227	0.073	2,791	0.077	3,545	0.080	3,465	0.081	4,000	0.082	4,965
1000	0.034	2,422	0.037	2,895	0.038	3,482	0.040	4,026	0.042	4,279	0.042	5,075
2000	0.030	2,370	0.034	2,819	0.036	3,320	0.038	3,861	0.041	3,878	0.042	4,492

Relative Bias is defined as

$$RB = (E(\hat{N}) - N)/N.$$

It is to measure how close the estimations are to the actual size of the data source. Standard Deviation (SD)is used to characterize the variations of estimations, i.e., how close the estimations are to the center of the estimations. It is defined as

$$SD = \sqrt{\frac{1}{i}\sum_{j=1}^{i}(\hat{N}_j - E(\hat{N}))^2}$$

When different data sources are involved, we use Relative Standard Deviation $RSD = SD/E(\hat{N})$ instead of SD for ease of comparison.

We carried out a simulation study that involves 30 combinations of five different values (10, 20, 100, 1000, and 2000) for k and six different values (2,3,4,5,6, and 7) for m/k. We report the case where N=1,000,000. Other values for N are also tested and similar results are observed.

For each combination, we run 100 simulations and obtain the relative bias RB and standard deviation SD. In the simulation, m number of random numbers in the range of 1 to N are generated, then the top k numbers are retained and recorded. The process repeats until the overlapping rate $OR = 1.1$. The sample size is roughly equivalent to $0.1N(k/m)$.

The results in Table 4 shows that all the estimations have a positive bias as expected, because documents d_i for $i > (k/m)N$ can still be captured although the probability is very small. The relative bias decreases when k becomes larger. When $k = 1000$, which is the limit set by many search engines including Google, the relative bias is smaller than 0.05. The variance is rather small and grows proportional to the value of m/k. Figure 3 visually depicts the estimations in one trial with more combinations of k and m/k. Overall, the simulation confirms that $\hat{N}_{0r}$ is a good estimator when k is not very small.

5. Model M_{hr}

When documents are retrieved by queries, their probabilities being matched are actually not equal. For example, larger documents will have higher proba-

bilities being matched by queries.

We use Figure 4 to explain the evolution of models from M_0 to M_{hr}. Each scatter plot records 600 captured documents, including duplicates, that are retrieved in a particular model. There are 1,000 documents, ordered by file size in decreasing order. The document with ID 1 is the largest file.

In model M_0, we randomly select the documents directly from the collection with uniform distribution. Hence every document has the same probability of being matched, and all the matched documents are returned. As Figure 4 (A) and (C) show, 600 captures are uniformly distributed.

Figure 4(B)is produced by model M_h, i.e., the probability of a document being matched is proportional to its file size, and all the matched documents are returned. It shows that the large documents, i.e., the documents with smaller document IDs, are retrieved more frequently over the querying process. Small documents, especially the documents with IDs closer to 1000, are not matched so often, but they are still possible being captured.

Figure 4(D) is produced by model M_{0r}, i.e., each document has the same probability of being matched, but only the top 10 documents are returned out of the 20 matches. As the figure shows, the first half of the documents have almost the same probability of being retrieved. Small documents are not likely to be obtained. As we analyzed in the previous section, when k is larger there will be a more clear-cut boundary around document id 500–there will be very few documents retrieved with id greater than 500.

Model M_{hr} is illustrated in Figure 4(E), where 20 documents are matched with probabilities according to their file sizes, and then the top 10 documents are returned. Unlike M_{0r}, the documents in M_{hr} have an unequal probability of being matched, even when their ID's are small. The histogram in Figure 4(F) clearly shows the distinction between M_{0r} and M_{hr}.

5.1. Model M_h

In real data source size estimation, documents will have varying probabilities of being captured. Based on Equation 4, we conjecture that there is also a fixed relation between P and OR in real data sources, with a modified equation as below:

$$P = 1 - \alpha OR^{\beta}, \tag{Hyp}$$

or

$$ln(1 - P) = ln\alpha + \beta \times lnOR$$

By running regression using four collections of newsgroups data described in section 5.3, we obtained $\hat{\alpha} = 1.001$, $\hat{\beta} = -1.132$. Here the R square is 0.875, which means that the regression fits the data well. Hence we derive Equation 12

$$P = 1 - OR^{-1.1} \tag{12}$$

and the estimator

$$\hat{N}_h = M/(1 - OR^{-1.1}). \tag{13}$$

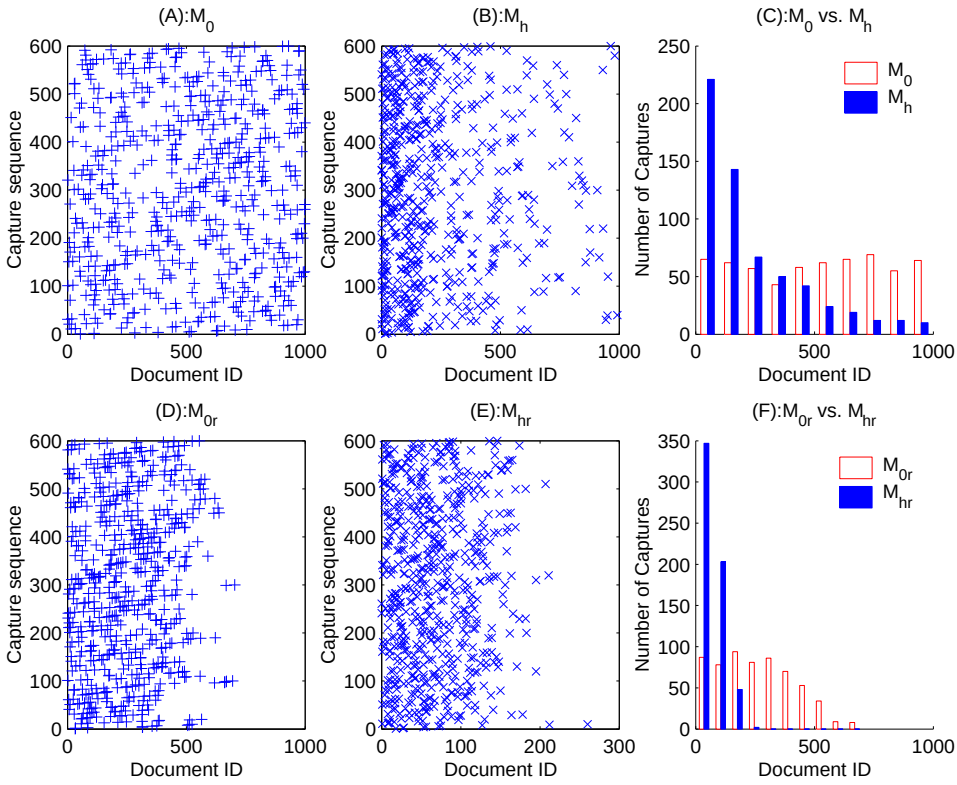

Figure 4: Comparison of models M_0, M_h, M_{0r}, and M_{hr}. 1000 documents are sorted according to their file size in decreasing order. 600 documents are selected in the four models, including the duplicates. $k = 10, m = 20$. Subplot M_0 shows that all the documents are retrieved uniformly. Subplot M_h shows that large documents are preferred, but most of the documents can be eventually sampled. Subplot M_{0r} exhibits a clear cut around the 500th document. Beyond this line there are almost no documents retrieved. M_{hr} is the compound of M_{0r} and M_h.

Table 5: Summary of test corpora. #words is the number of unique words that appear in Webster dictionary.

Corpus name	docs	Size in MB	Avg size(KB)	mean #words	SD of #words
Wiki	1,475,022	1950	1.35	284	285
Gov2_1	1,077,019	5420	5.15	396	409
Gov2_2	1,075,270	5241	4.99	389	407
NG_1	1,372,911	1023	0.76	294	223
Reuters	806,791	666	0.85	125	82

5.2. Model M_{hr}

In model M_{hr}, m documents are matched with unequal probability. With the understanding that a ranked data source will expose only the top $(k/m)N$ number of documents, the estimator for M_{hr} is $(m/k)\hat{N}_h$. While it would be ideal to select queries so that their document frequencies are equal to a fixed number m, in practice it would not be easy to collect such queries. Hence we need to allow some variations of document frequencies of the queries. Suppose that t number of queries are sent to the data source, each matches with m_i number of documents, where $1 \leq i \leq t$. We define the overflow rate as

$$\mathcal{OF} = \sum_{i=1}^{t} m_i/(kt). \tag{14}$$

Then the estimator we use is

$$\begin{aligned} \hat{N}_{hr} &= \mathcal{OF} \times \hat{N}_h \\ &= \mathcal{OF} \times M/(1 - OR^{-1.1}) \end{aligned} \tag{15}$$

5.3. Experiment

5.3.1. Data

We run our experiments on a variety of data collected from various domains. The corpora are Reuters, Gov2, Wikipedia, and Newsgroups, which are summarized in Table 5. We also produce the log-log plot of the file size distributions in Figure 5. These are standard test data that are used by many researchers in information retrieval. *Wikipedia* is the corpus provided by wikipedia.org which contains 1.4 millions of English documents. *Gov*2 is a TREC test data collected from .gov domain during 2004, which contains 25 million documents. We used two subsets of the data for efficiency consideration. *Newsgroups* corpus includes posts in various newsgroups. *Reuters* is a TREC data set that contains 806,790 news stories in English. The corpora are indexed using Lucene [20]. We do not carry out the experiments on real deep web data sources because their actual size is unknown, hence the evaluation of the methods will not be accurate.

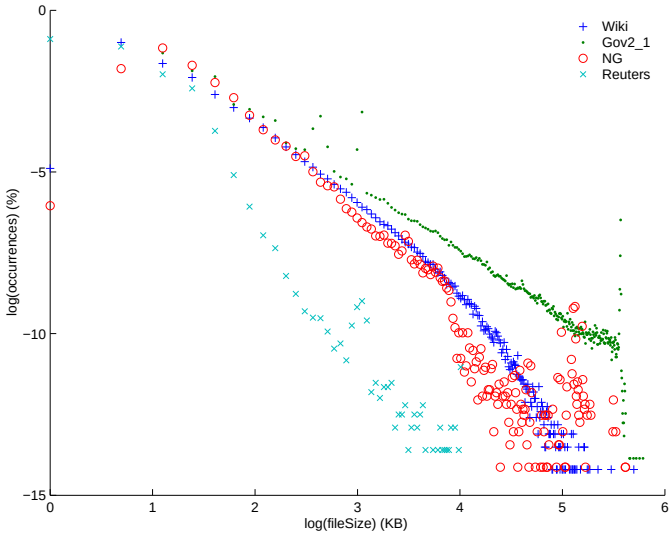

Figure 5: File size distributions of the four corpora.

5.3.2. Experiment

We conduct experiments on 30 combinations of five corpora and six overflow rates. 5,000 words are randomly selected from Webster dictionary which consists of about 60,000 terms. More precisely, let each term be represented by a unique number ranging between 1 and 60,000. The randomness is implemented by generating 5,000 random numbers with uniform distribution between 1 and 60,000.

Among the 5000 words we select the queries whose document frequencies vary within the range of 3500 and 4500, i.e., $m = 3500 \sim 4500$. Different overflow rates are obtained by setting varying k, i.e., the return limit k is set as 100, 200, 400, 800, 1600, and 3200. Note that although in each trial 5000 queries are randomly selected, the average number of queries sent to a data source is in the range of 39 to 146 as shown in Table 6.

For each combination we run 20 trials. Different trials are obtained by selecting randomly another set of 5000 queries from Webster. From the 20 runs we obtain RB and RSD. Table 6 is the overview of the experiment result. The standard deviations for all combinations are very small. The bias for various overflow rates are rather stable, but varies from corpus to corpus. In general it works well, and can be explained by the file size distributions depicted in Figure 5– all the corpora file sizes follow the power law with similar exponents, hence the degree of matching heterogeneity is similar. On the other hand, Gov2 has a heavier tail, i.e., there are more large documents than other corpora. It implies that the Gov2 has a higher degree of matching heterogeneity, and drifts farther away from model M_0. This is why Gov2 exhibits a larger negative bias.

Table 7 tabulates the details for one trial of the estimation. Column t is the

Table 6: $\hat{N}_{hr}$ for combinations of corpora and overflow rates. $m = 3500 \sim 4500$. RB and RSD are obtained from 20 runs.

	k											
	100		200		400		800		1600		3200	
corpus	RB	RSD	RB	RSD	RB	RSD	RB	RSD	RB	RSD	RB	RSD
Wiki	-0.21	0.04	-0.22	0.04	-0.23	0.03	-0.25	0.04	-0.24	0.03	-0.23	0.02
GOV2_1	-0.37	0.03	-0.38	0.05	-0.39	0.04	-0.39	0.04	-0.39	0.03	-0.47	0.03
GOV2_2	-0.35	0.05	-0.37	0.03	-0.41	0.04	-0.40	0.07	-0.42	0.04	-0.46	0.03
NG	0.29	0.11	0.30	0.07	0.21	0.06	0.11	0.08	-0.05	0.09	-0.12	0.05
Reuters	0.18	0.08	0.13	0.10	0.06	0.08	0.06	0.09	0.07	0.07	-0.03	0.09

number of queries issued for the estimation. We can see that the numbers of queries are below 150, which indicate that the estimation process is very efficient. The column n is the sample size, i.e., the total number of documents retrieved. It is dependent on the value of k– a smaller k will result in a smaller sample size. When m/k is large, sample size can be reduced dramatically. Column M is the number of unique documents that are retrieved. With M and n, we can calculate the overlapping rate OR, and the estimations by various estimators $\hat{N}_0, \hat{N}_h$, and $\hat{N}_{hr}$. $\hat{N}_{sReg}$ is the estimator developed in [34] and will be compared in the next section.

From the table it shows that both $\hat{N}_0$ and $\hat{N}_h$ become closer to real value when k increases, while $\hat{N}_0$ is further away from the truth because it does not consider the unequal catching probability of a document being matched. When k (hence k/m since here m is a fixed value) is small, even $\hat{N}_h$ is very far away from the real value. For example, when $k = 100$, $\hat{N}_h$ for Wiki corpus is only 31,965, while the true value is 1.4M. By using the overflow rate, $\hat{N}_{hr}$ gives a rather good estimation.

5.4. Comparison

Shokouhi et al's $\hat{N}_{sReg}$ estimator [34] is the closest to ours in that their method is also based on multiple capture-recapture method. In addition, they also use document ids only, hence saving the effort of document downloading and parsing. They start with the traditional Schumacher estimator $\hat{N}_s$ (Equation 3) [35], whose estimation result is very close to $\hat{N}_0$.

Then Shokouhi et al. correct the bias of $\hat{N}_s$ method using regression. They conjecture that there is a fixed relationship between the initial estimation, which is obtained by $\hat{N}_s$, and the actual data size. Based on this hypothesis, they use a training data set to obtain an equation between the initial estimation and the actual size using non-linear regression.

In the last column of Table 7, we list the estimation by $\hat{N}_{sReg}$. While it works fine when the return limit k is around 1000, it introduces very large bias when k varies. For example, for Wiki corpus, $\hat{N}_{sReg}$ ranges between 29, 415 and 6,030,502.

Both our method and $\hat{N}_{sReg}$ method start with M_0, albeit the estimators are different. In fact, $\hat{N}_0$ and $\hat{N}_{CH}$ produce very close estimations. What different

Table 7: Details of one trial. k is the return limit, t is the number of queries issued, n is the sample size, M is the number of unique documents retrieved. $m = 3500 \sim 4500$.

	k	t	n	M	OR	$\hat{N}_0$	$\hat{N}_h$	$\hat{N}_{hr}$	$\hat{N}_{sReg}$
Wiki	100	134	13,400	9661	1.39	19,441	31,965	1,278,600	29,415
	200	146	29,200	20,470	1.43	38,937	63,289	1,265,780	86,412
	400	133	53,200	38,502	1.38	78,114	128,637	1,286,370	256,354
	800	138	110,400	78,437	1.41	153,143	250,281	1,251,405	719,154
	1,600	146	233,600	162,607	1.44	305,252	494,735	1,236,837	2,102,289
	3,200	143	457,600	318,910	1.43	599,985	972,888	1,216,110	6,030,502
GOV2_1	100	91	9100	6271	1.45	11,560	18,660	746,400	12,914
	200	97	19,400	13010	1.49	22,909	36,581	731,620	37,147
	400	85	34,000	23691	1.44	44,556	72,244	722,440	106,658
	800	107	856,00	54135	1.58	87,602	136,737	683,685	298,941
	1,600	90	144,000	97,212	1.48	173,029	277,014	692,535	875,090
	3,200	86	275,200	182,098	1.51	314,030	498,796	623,495	2,172,136
NG	100	128	12,800	10232	1.25	27,274	46,865	1,874,600	49,574
	200	129	25,800	20483	1.26	53,330	91,361	1,827,220	143,095
	400	105	42,000	33759	1.24	91,764	158,062	1,580,620	336,603
	800	118	94,400	74161	1.27	186,549	318,115	1,590,575	1,003,814
	1,600	129	206,400	154214	1.34	336,868	562,203	1,405,507	2,523,227
	3,200	122	390,400	293768	1.33	653,325	1,093,626	1,367,032	6,951,760
Reuters	100	54	5,400	4,498	1.20	14,111	24,697	987,880	18,045
	200	40	8,000	6,999	1.14	28,595	51,183	1,023,660	57,214
	400	51	20,400	17,051	1.20	54,337	95,245	952,450	145,946
	800	39	31,200	26,296	1.19	87,160	153,356	766,780	315,683
	1,600	61	97,600	78,449	1.24	213,237	367,295	918,237	1,258,096
	3,200	58	185,600	147,453	1.26	384,814	659,437	824,296	3,082,789

is the next step. $\hat{N}_{sReg}$ is solely dependent on $\hat{N}_{CH}$, regardless of the return limit k and the heterogeneity of the matching probability. In contrast, our method adjust $\hat{N}_0$ to $\hat{N}_h$ first, so that the heterogeneity of matching probability is accounted for. Next, it is extended to $\hat{N}_{hr}$ to accommodate the stratified data layers.

Compared with the sampling based approach, our method has the following advantages:

1. Sampling approach in general needs to send more queries and retrieve more documents, because it rejects some queries or documents. In many cases the rejected queries and documents can be very large. For example, to obtain one random documents, thousands of documents may need to be downloaded and analyzed [37].
2. Sampling approach needs downloading and lexical analysis of the documents, in order to know the size and weight [4] of the documents. In many applications, downloading may be impossible. For example, Amazon book store will allow you have access to the basic information of the book, not the entire book. For this type of data sources, sampling based approach will not able to work.
3. In order to guarantee that queries do not overflow, sampling based approach need to use rare words as queries or conjunctive queries. Rare

queries also tend to underestimate the collection size, hence introduce another bias. In addition, many rare queries do not match any documents, thereby add more estimation cost.

6. Conclusions

This paper proposes an efficient estimation method based capture recapture method. Data sources vary in several aspects, such as whether they are ranked, what is the return limit, and whether they favor large documents. When the assumptions of a data source change, estimator should also differ. Hence this paper abstracts the types of data sources into several models, including models M_0, M_h, M_{0r}, and M_{hr}. While M_0 and M_h are widely studied in ecology, ranked models M_{0r} and M_{hr} proposed in this paper are unique in the area of data source size estimation.

In order to derive the estimator for ranked data sources, we start from an over simplified model M_0 where each document has the same probability of being retrieved. From M_0, we develop model M_{0r}, where each document has the same probability of being matched, but only top k documents will be returned. For model M_{0r}, we give the estimator $\hat{N}_{0r}$ which is analytically derived and empirically verified. We also identify the condition under which it works, i.e., the return limit k should be greater than 100.

Based on M_{0r}, we develop model M_{hr} for real data sources where the matching probability varies from document to document. Although the result is empirical, we have tested extensively on a variety of corpora. Also, it can be explained by the file size distributions of the corpora.

In addition to the accuracy of the estimator, our method is very efficient– only about 100 random queries are needed to determine the size of a data source. In particular, our method only needs to know the document ID in order to decide whether there are overlapping documents. In the contrast, sampling based methods need to download and lexically analyze many documents [4][11]. Our approach is also more efficient than other estimator based methods. For example, Shokouhi et al.'s method need to fire 5,000 queries [34].

The result of this paper can be also applied to deep web crawling [29][26]. When selecting the queries to crawl a deep web, one may be tempted to select those with large document frequencies. However, for ranked data sources, those popular queries will induce higher overflow rate when the return limit k is fixed, hence will scoop only the data in the top layer. According to the results in this paper, the queries should be selected so that overflow rate is minimized.

7. Acknowledgements

The research is supported by NSERC (Natural Sciences and Engineering Research Council Canada). The author would like thank the reviewers for their detailed comments, Dr. Dingding Li for very insightful discussions, and Jie Liang for preparing the data in the experiments.

[1] SC Amstrup, TL McDonald, BFJ Manly, Handbook of Capture-Recapture Analysis, Princeton University Press, 2005.

[2] L.Barbosa and J. Freire, Siphoning hidden-web data through keyword-based interfaces, SBBD, 2004.

[3] Anagnostopoulos, I. and Stavropoulos, P. 2006. Adopting Wildlife Experiments for Web Evolution Estimations: The Role of an AI Web Page Classifier. In Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence (December 18 - 22, 2006). Web Intelligence. IEEE Computer Society, 897-901.

[4] Bar-Yossef, Z. and Gurevich, M. 2006. Random sampling from a search engine's index. WWW , 2006. 367-376.

[5] Bar-Yossef, Z. and Gurevich, M. Efficient search engine measurements. WWW , 2007. 401-410.

[6] Ziv Bar-Yossef, Maxim Gurevich: Random sampling from a search engine's index. J. ACM 55(5) (2008).

[7] Michael K. Bergman, The Deep Web: Surfacing Hidden Value, The Journal of Electronic Publishing 7 (1). 2001.

[8] Bharat, K. and Broder, A. A technique for measuring the relative size and overlap of public Web search engines. WWW 1998, 379-388.

[9] Igor A. Bolshakov, Sofia N. Galicia-Haro, Can We Correctly Estimate the Total Number of Pages in Google for a Specific Language? CICLing 2003: 415-419.

[10] Paul Bourret: How to Estimate the Sizes of Domains. Inf. Process. Lett. 19(5): 237-243 (1984)

[11] Broder, A., Fontura, M., Josifovski, V., Kumar, R., Motwani, R., Nabar, S., Panigrahy, R., Tomkins, A., and Xu, Y. 2006. Estimating corpus size via queries. CIKM '06. 594-603.

[12] Callan,J. and M. Connell, Query-Based Sampling of Text Databases. ACM Transactions on Information Systems, 2001. 97-130.

[13] James Caverlee, Ling Liu, and David Buttler, Probe, Cluster, and Discover: Focused Extraction of QA-Pagelets from the Deep Web, ICDE 2004. 103-114.

[14] Chao, A. and Lee, S-M. (1992). Estimating the number of classes via sample coverage. Journal of American Statistical Association, 87, 210-217.

[15] V. Crescenzi, G. Mecca, P. Merialdo, RoadRunner: towards automatic data extraction from large web sites, VLDB J. 2001. pp. 109-118.

[16] Darroch, J. N., The Multiple-recapture Census: I . Estimation of a Closed Population, Biometrika, Vol. 45, No. 3/4 (Dec., 1958), pp. 343-359.

[17] A. Dobra and S. Fienberg. How large is the World Wide Web? Web Dynamics, 23-44, 2004.

[18] Antonio Gulli, A. Signorini: The indexable web is more than 11.5 billion pages. WWW 2005. 902-903.

[19] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, Lynne Stokes: Sampling-Based Estimation of the Number of Distinct Values of an Attribute. VLDB 1995: 311-322.

[20] Hatcher, E. and O. Gospodnetic, Lucene in Action, Manning Publications, 2004.

[21] Ipeirotis, P. G., Gravano, L., and Sahami, M. 2001. Probe, count, and classify: categorizing hidden web databases. SIGMOD 2001.

[22] Stephen W. Liddle, David W. Embley, Del T. Scott, Sai Ho Yau, Extracting Data behind Web Forms, Advanced Conceptual Modeling Techniques, 2002. 402-413.

[23] Liu, K., Yu, C., and Meng, W. Discovering the representative of a search engine. CIKM '02. 652-654.

[24] Jianguo Lu, Efficient estimation of the size of text deep web data source, CIKM 2008. 1485-1486.

[25] Jianguo Lu, Dingding Li, Estimating Deep Web Data Source Size by Capture-Recapture Method, Information Retrieval, Springer 2010. Vol 13(1) 2010. pp. 70-95.

[26] Jianguo Lu, Yan Wang, Jie Liang, Jessica Chen, Jiming Liu, An approach to deep web crawling by sampling, WI 2008.

[27] Madhavan, Jayant; David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, Alon Halevy. Google's Deep-Web Crawl. VLDB 2008. pp. 1241-1252.

[28] Nelson, M. L., Smith, J. A., and del Campo, I. G. Efficient, automatic web resource harvesting. WIDM '06, 43-50.

[29] Alexandros Ntoulas, Petros Zerfos, and Junghoo Cho, Downloading Textual Hidden Web Content through Keyword Queries. JCDL, 2005. 100-109.

[30] Pollock, K. H., J. D. Nichols, C. Brownie, and J. E. Hines. 1990. Statistical inference for captureCrecapture experiments. Wildlife Monographs 107.

[31] S. Raghavan, H. Garcia-Molina. Crawling the Hidden Web, VLDB 2001.

[32] Scott, H. and Wohlin, C. 2008. Capture-recapture in software unit testing: a case study. In Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement (Kaiserslautern, Germany, October 09 - 10, 2008). ESEM '08. ACM, New York, NY, 32-40.

[33] Denis Shestakov, Sourav S. Bhowmick and Ee-Peng Lim, DEQUE: querying the deep web, Journal of Data Knowl. Eng., 52(3), 2005. 273-311.

[34] M Shokouhi, J Zobel, F Scholer, SMM Tahaghoghi, Capturing collection size for distributed non-cooperative retrieval, SIGIR 2006. 316-323.

[35] Schumacher, F. X. Eschmeyer, R. W. (1943). The estimation of fish populations in lakes or ponds. J. Tenn. Acad. Sci.18, 228-249.

[36] Si, L. and Callan, J. 2003. Relevant document distribution estimation method for resource selection. SIGIR 2003.

[37] Paul Thomas, David Hawking, Evaluating Sampling Methods for Uncooperative Collections, SIGIR, 2007.

[38] Paul Thomas, Generalising multiple capture-recapture to non-uniform sample sizes. SIGIR 2008. pp. 839-840.

[39] Weaver, R. and Collins, M. P. 2007. Fishing for phishes: applying capture-recapture methods to estimate phishing populations. In Proceedings of the Anti-Phishing Working Groups 2nd Annual Ecrime Researchers Summit (Pittsburgh, Pennsylvania, October 04 - 05, 2007). eCrime 2007, vol. 269. ACM, New York, NY, pp. 14-25.

[40] S. Wu, F. Gibb, and F. Crestani. Experiments with document archive size detection. 25th European Conference on IR Research, 2003. pp. 294–304.

[41] Ping Wu, Ji-Rong Wen, Huan Liu, Wei-Ying Ma, Query Selection Techniques for Efficient Crawling of Structured Web Sources, ICDE, 2006. pp. 47-56.

[42] Xu, J., Wu, S., and Li, X. 2007. Estimating collection size with logistic regression. SIGIR 2007. pp. 789-790.

[43] F Yan, WC Hou, Z Jiang, C Luo, Q Zhu, Selectivity estimation of range queries based on data density approximation via cosine series, Data and Knowledge Engineering, Volume 63, Issue 3, December 2007, Pages 855-878