Crowdsourced Enumeration Queries

Beth Trushkowsky, Tim Kraska, Michael J. Franklin, Purnamrita Sarkar
AMPLab, UC Berkeley, United States
{trush, kraska, franklin, psarkar}@cs.berkeley.edu

***Abstract*— Hybrid human/computer database systems promise to greatly expand the usefulness of query processing by incorporating the crowd for data gathering and other tasks. Such systems raise many implementation questions. Perhaps the most fundamental question is that the closed world assumption underlying relational query semantics does not hold in such systems. As a consequence the meaning of even simple queries can be called into question. Furthermore, query progress monitoring becomes difficult due to non-uniformities in the arrival of crowdsourced data and peculiarities of how people work in crowdsourcing systems. To address these issues, we develop statistical tools that enable users and systems developers to reason about query completeness. These tools can also help drive query execution and crowdsourcing strategies. We evaluate our techniques using experiments on a popular crowdsourcing platform.**

I. Introduction

Advances in machine learning, natural language processing, image understanding, etc. continue to expand the range of problems that can be addressed by computers. But despite these advances, people still outperform state-of-the-art algorithms for many data-intensive tasks. Such tasks typically involve ambiguity, deep understanding of language or context, or subjective reasoning.

Crowdsourcing has emerged as a paradigm for leveraging human intelligence and activity at large scale. Popular crowdsourcing platforms such as Amazon Mechanical Turk (AMT) provide access to hundreds of thousands of human workers via programmatic interfaces (APIs). These APIs provide an intriguing new opportunity, namely, to create hybrid human/computer systems for data-intensive applications. Such systems, could, to quote J.C.R. Licklider's famous 1960 prediction for man-computer symbiosis, "...process data in a way not approached by the information-handling machines we know today." [1].

A. Query Processing with Crowds

Recently, a number of projects have begun to explore the potential of hybrid human/computer systems for database query processing. These include CrowdDB [2], Qurk [3], and sCOOP [4]. In these systems, human workers can perform query operations such as subjective comparisons, fuzzy matching for predicates and joins, entity resolution, etc. As shown in [2], these simple extensions can greatly extend the usefulness of a query processing system.

Of course, many challenges arise when adding people to query processing, due to the peculiarities in latency, cost, quality and predictability of human workers. For example, data obtained from the crowd must be validated, spelling mistakes must be fixed, duplicates must be removed, etc. Similar issues arise in data ingest for traditional database systems through ETL (Extract, Transform and Load) and data integration, but techniques have also been developed specifically for crowdsourced input [5], [6], [7], [8].

The above concerns, while both interesting and important, are not the focus of this paper. Rather, we believe that there are more fundamental issues at play in such hybrid systems. Specifically, when the crowd can augment the data in the database to help answer a query, the traditional *closed-world assumption* on which relational database query processing is based, no longer holds. This fundamental change calls into question the basic meaning of queries and query results in a hybrid human/computer database system.

B. Can You Really Get it All?

In this paper, we consider one of the most basic RDBMS operation, namely, scanning a single table with predicates. Consider, for example, a SQL query to list all restaurants in San Francisco serving scallops: `SELECT * FROM RESTAURANTS WHERE CITY = 'San Francisco' and DISH = 'Scallops'`. In a traditional RDBMS there is a single correct answer for this query, and it can be obtained by scanning the table, filtering the records, and returning all matching records of the table. This approach works even for relations that are in reality unbounded, because the closed world assumption dictates that any records not present in the database at query execution time do not exist. Of course, such limitations can be a source of frustration for users trying to obtain useful real-world information from database systems.

In contrast, in a crowdsourced system like CrowdDB, once the records in the stored table are exhausted, jobs can be sent to the crowd asking for additional records. The question then becomes: when is the query result set complete? Crowdsourced queries can be inherently fuzzy or have unbounded result sets, with tuples scattered over the web or only in human minds. For example, consider a query for a list of graduating Ph.D. students currently on the job market, or companies in California interested in green technology. Such queries are often considered one of the main use cases for crowd-enabled database systems, as they reside in the sweet spot between too much work for the user but not executed often enough to justify a complex machine learning solution.

In this paper we address the question of "How should users think about enumeration queries in the open world of a crowdsourced database system?". We develop statistical

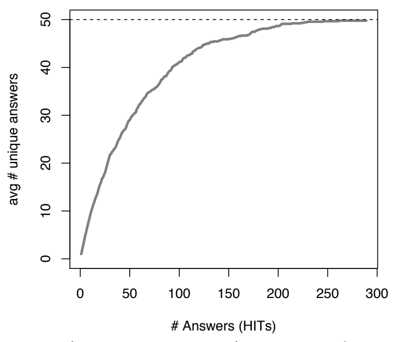


Fig. 1. States experiments: average unique vs. total number of answers

tools that enable users to reason about tradeoffs between time/cost and completeness, and that can be used to drive query execution and crowdsourcing strategies.

C. *Counting Species*

The key idea of our technique is to use the arrival rate of new answers from the crowd to reason about the completeness of the query. Consider the execution of a "`SELECT *`" query in a crowdsourced database system where workers are asked to provide individual records of the table. For example, one could query for the names of the 50 US states using a microtask crowdsourcing platform like AMT by generating HITs (i.e., Human Intelligence Tasks) that would have workers provide the name of one or more states. As workers return results, the system collects the answers, keeping a list of the unique answers (suitably cleansed) as they arrive.

Figure 1 shows the results of running that query, with the number of unique answers received shown on the vertical axis, and the total number of answers received on the x-axis. As would be expected, initially there is a high rate of arrival for previously unseen answers, but as the query progresses (and more answers have been seen) the arrival rate of new answers begins to taper off, until the full population (i.e., the 50 states, in this case) has been identified.

This behavior is well-known in fields such as biology and statistics, where this type of figure is known as the *Species Accumulation Curve* (SAC) [9]. Imagine you were trying to count the number of unique species of animals on an island by putting out traps overnight, identifying the unique species found in the traps the next morning, releasing the animals and repeating this daily. By observing the rate at which new species are identified over time, you can begin to infer how close to the true number of species you are. We can use similar reasoning to help understand the execution of set enumeration queries in a crowdsourced query processor.

D. *Overview of the Paper*

In this paper, we apply species estimation techniques from the statistics and biology literature to understand and manage the execution of set enumeration queries in crowdsourced database systems. We find that while the classical theory provides the key to understanding the meaning of such queries, there are certain peculiarities in the behavior of microtask crowdsourcing workers that require us to develop new methods to improve the accuracy of cardinality estimation in this

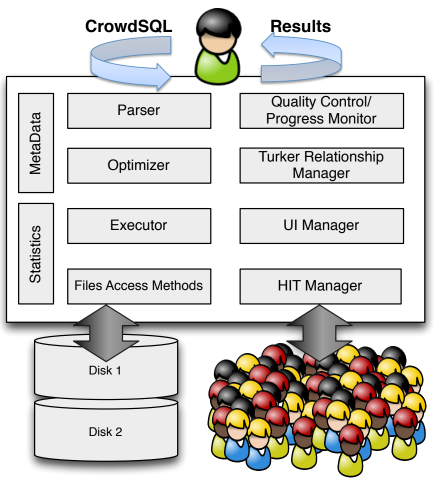


Fig. 2. CrowdDB Architecture

environment. We also describe methods to leverage these techniques to help users make intelligent tradeoffs between time/cost and completeness. These techniques extend beyond crowdsourced databases and, for example, can help to estimate the completeness of deep-web queries.

To summarize, we make the following contributions:

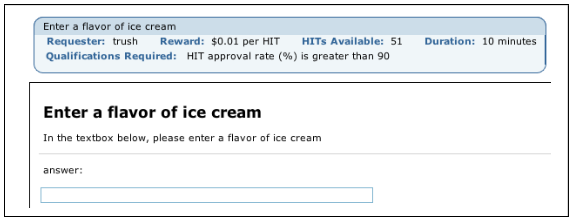
- We formalize the process of crowdsourced set enumeration and describe how it violates statistical fundamental assumptions of existing species estimation techniques.
- We develop a technique to estimate result cardinality and query progress in the presence of crowd-specific behaviors.
- We devise a pay-as-you-go approach to allow informed decisions about the cost/completeness tradeoff, we well as a technique to determine if data scraping could be applied.
- We examine the effectiveness of our techniques via experiments using Amazon Mechanical Turk (AMT).

The paper is organized as follows: In Section II we give background on the CrowdDB system. Section III describes how crowd-specific behaviors break assumptions on which species estimation algorithms are based. In Section IV we develop techniques to improve the estimation in the presence of crowd-specific behavior, whereas Section V discusses a heuristic to detect when alternate data-gathering techniques could be used. Section VI introduces a pay-as-you-go technique. Section VII covers related work and in Section VIII we conclude.

II. Background: CrowdDB

CrowdDB is a hybrid human-machine database system that uses human input to process queries. CrowdDB currently supports two crowdsourcing platforms: AMT and our own mobile platform [10]. We focus on AMT in this paper, the leading platform for so-called microtasks. Microtasks, also called Human Intelligence Tasks (HITs) in AMT, usually do not require any special training and do not take more than a few minutes to complete. AMT provides a marketplace for microtasks that allows requesters to post HITs and workers to search for and work on HITs for a small reward, typically a few cents each.

Figure 2 shows the architecture of CrowdDB. CrowdDB incorporates traditional query compilation, optimization and



Enter a flavor of ice cream

Requester: trush Reward: $0.01 per HIT HITs Available: 51 Duration: 10 minutes

Qualifications Required: HIT approval rate (%) is greater than 90

Enter a flavor of ice cream

In the textbox below, please enter a flavor of ice cream

answer:

Fig. 3. Ice cream flavors task UI on AMT

execution components, which are extended to cope with human-generated input. In addition the system is extended with crowd-specific components, such as a user interface (UI) manager and quality control/progress monitor. Users issue queries using CrowdSQL, an extension of standard SQL. CrowdDB automatically generates UIs as HTML forms based on the `CROWD` annotations and optional free-text annotations of columns and tables in the schema. Figure 3 shows an example HTML-based UI that would be presented to a worker for the following crowd table definition:

```
CREATE CROWD TABLE ice_cream_flavor {
    name VARCHAR PRIMARY KEY
}
```

Although CrowdDB supports alternate user interfaces (e.g., showing previously received answers), this paper focuses on a pure form of the set enumeration question. The use of alternative UIs is the subject of future work.

During query processing, the system automatically posts one or more HITs using the AMT web service API and collects the answers as they arrive. After receiving the answers, CrowdDB performs simple quality control using quorum votes before it passes the answers to the query execution engine. Finally, the system continuously updates the query result and estimates the quality of the current result based on the new answers. The user may thus stop the query as soon as the quality is sufficient or intervene if a problem is detected. More details about the CrowdDB components and query execution are given in [2]. This paper focuses on the quality control and progress component that allows the user to continuously reason about query completeness and cost.

III. A Model and Analysis of Progress Estimation for Crowdsourced Enumerations

To evaluate progress as answers are arriving, the system needs an estimate of the result set's cardinality in order to calculate the percentage complete. Species estimation algorithms from statistics and biology literature tackle a similar goal: an estimate of the number of distinct species is determined using observations of species in the locale of interest. These techniques are also used in traditional database systems to inform query optimization of large tables [11]. In this section, we describe our observations of how the crowd answers set-enumeration queries and why existing estimation techniques yield inaccurate estimates. We also present a model for crowdsourced enumerations and list the requirements for human-tolerant cardinality estimators.

A. The Problem with Existing Estimators

Various techniques have been devised in biology to estimate the number of species [12], [13] as well as in the database community to estimate the number of distinct values in a table [11]. They all operate similarly: a sample is drawn at random from a population (e.g., the entire table) and based on the frequency of observed items (distinct values), the number of unobserved items (number of missing distinct values) is estimated. The techniques differ most notably in their assumptions, in particular that distinct value estimation techniques assume that the population (i.e., table) size is known. Unfortunately, knowledge of the population size is only possible in the closed world; in systems with crowdsourced enumerations, records can be acquired on-demand, thus the table size is potentially infinite. We focus the remaining discussion on species estimators suitable for the open world because they allow for an infinite population.

To gain an understanding of the crowd's ability to answer set enumeration queries and its impact on existing estimation techniques, we crowdsourced the elements of sets for which the true cardinality is known, using the UI exemplified in Figure 3. We use the open-world-safe estimator "Chao92" [14] as it is widely used in the species estimation literature [15].[1] Figure 4 shows the observed Chao92 estimate ("actual") evaluated as answers arrive in one AMT experiment in which we crowdsourced the names of the 192 United Nations (UN) member countries and compares it to the expected behavior using simulation with the empirical data distribution derived from all our UN experiment runs. We focus on a single representative experiment rather than an average over multiple runs to investigate the behavior a user would observe; averaging can also disguise the effects described next.

Note in Figure 4 that the value of the estimate begins approaching the true value of 192, however it then significantly *overestimates* the true value for most of the remaining time of the experiment. This is surprising as our simulation shows that the estimate should become more accurate and stable as it receives more data ("expected" in Figure 4). As it turns out, the way in which crowd workers each provide their answers deeply impacts the behavior of an estimation algorithm. For example, in five runs of the UN experiment, we observed various trends in how the crowd responds to a set enumeration query. A worker could enumerate the UN countries by traversing an alphabetical list, starting with Afghanistan. However, it was not uncommon for workers to begin their answer sequence with a few countries they knew of (e.g., United States, India, Pakistan, China, etc.), or to provide a completely non-alphabetical sequence. We even observed alphabetical traversals that began at the end or in the middle of the alphabet! In general, people may use different internal biases or techniques for finding items in the set (we discuss full list traversals in Section V). We also noticed that individual

[1]We experimented with various other estimators for the open-world such as "Chao84"[16], "Jackknife"[17], and "uniform maximum-likelihood"[14] and also found Chao92 to be superior.

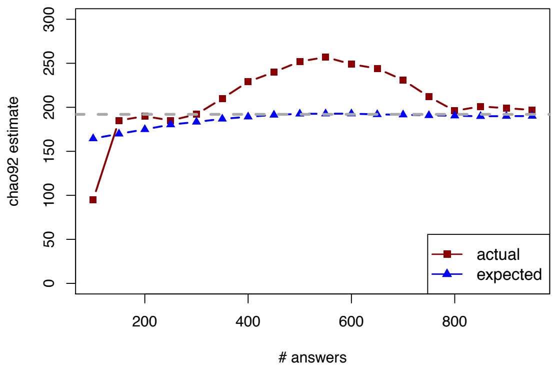
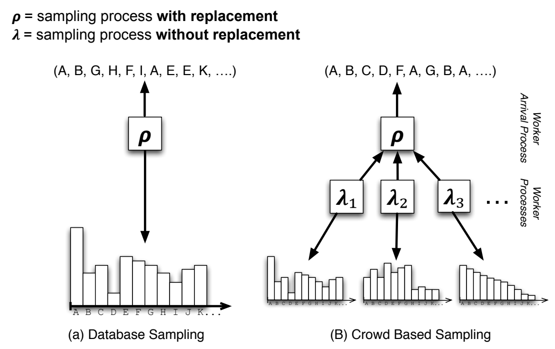


Fig. 4. Estimated Cardinality


ρ = sampling process with replacement
λ = sampling process without replacement
(A, B, G, H, F, I, A, E, E, K, ...)
(A, B, C, D, F, A, G, B, A, ...)
Worker Arrival Process
Worker Processes
(a) Database Sampling
(B) Crowd Based Sampling
Fig. 5. Sampling Process

workers complete different amounts of work and arrive/depart from the experiment at different points in time.

The next subsection formalizes a model of how answers arrive from the crowd in response to a set enumeration query, as well as a description of how crowd behaviors impact the sample of answers received. We then use simulation to demonstrate the principles of how these behaviors play off one another and thereby influence an estimation algorithm.

B. A Model for Human Enumerations

Species estimation algorithms assume a with-replacement sample from some unknown distribution describing item likelihoods (visualized in Figure 5(a)). The order in which elements of the sample arrive is irrelevant in this context.

After analyzing the crowdsourced enumerations, for example in the previously mentioned UN experiment, we found that this assumption does not hold for crowdsourced sets. In contrast to with-replacement samples, workers provide answers from an underlying distribution *without* replacement. Furthermore, workers might sample from different underlying distributions (e.g., one might provide answers alphabetically, while another worker provides answers in a more random order).

This process of sampling significantly differs from what traditional estimators assume, and it can be represented as a two-layer sampling process as shown in Figure 5(b). The bottom layer consists of many sampling processes, each corresponding to one worker, that sample from some data distribution *without replacement*. The top layer processes samples *with* replacement from the set of the bottom-layer processes (i.e., workers). Thus, the ordered stream of answers from the crowd represents a with-replacement sampling amongst workers who are each sampling a data distribution without replacement.

C. The Impact of Humans

The impact of the two-layer sampling process on the estimation can vary significantly based on the parameterization of the process (e.g., the number of worker processes, different underlying distributions, etc.). In this section, we study the impact of different parameterizations, define when it is actually possible to make a completeness estimation as well as the requirements for an estimator considering human behavior.

1) Sampling Without Replacement: When a worker submits multiple items for a set enumeration query, each answer is different from his previous ones. In other words, individuals are sampling without replacement from some underlying distribution that describes the likelihood of selecting each answer. Of course, this behavior is beneficial with respect to the goal of acquiring all the items in the set, as low-probability items become more likely after the high-probability items have already been provided by that worker (we do not pay for duplicated work from a single worker). A negative side effect of workers sampling without replacement is that the estimation algorithm receives less information about the relative frequency of items, and thus the skew, of the underlying data distribution; having knowledge of the skew is a requirement for a good estimate.

2) Worker skew: On crowdsourcing platforms like AMT, it is common that some workers complete many more HITs than others. This skew in relative worker HIT completion has been labeled the "streakers vs. samplers" effect [18]. In the two-layer sampling process, worker skew dictates which worker supplies the next answer; streakers are chosen with higher frequency. High worker skew can cause the arrival rate of unique answers to be even more rapid than that caused by sampling without replacement alone, causing the estimator to over-predict. The reasoning is intuitive: if one worker gets to provide a majority of the answers, and he provides only answers he has not yet given, then a majority of the total answer set will be made up of his unique answers.

Furthermore, in an extreme scenario in which one worker provides all answers, the two-layer process reduces to one process sampling from one underlying distribution without replacement. In this case, completeness estimation becomes impossible because no inference can be made regarding the underlying distribution. Another extreme is if an infinite number of "samplers" would provide one answer each using the same underlying distribution, the resulting sample would correspond to the original scenario of sampling with replacement (Figure 5(a)).

The latter is the reason why it is still possible to make estimations even in the presence of human-generated set enumerations. Figure 6(a) shows the impact of having more workers on the averaged Chao92 estimates with 100 simulation runs using a uniform data distribution over 200 items. As expected,

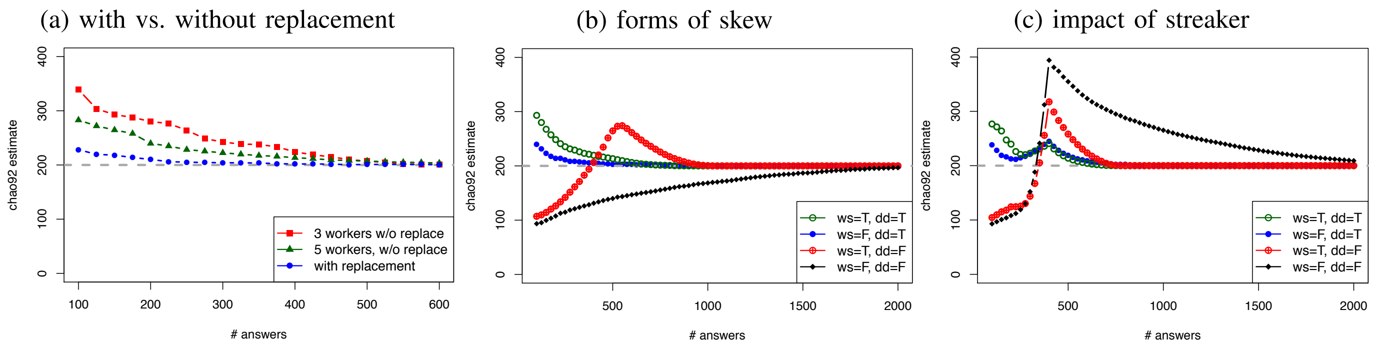


Fig. 6. Cardinality estimation simulations illustrating the impact of worker behaviors

the with-replacement sample overestimates slightly because of the uniform data distribution, but quickly approaches the true value of 200. The without-replacement samples overestimate even more and remain in that state for longer.

D. Different and Skewed Data Distributions

Individual workers also may be drawing their answers from different data distributions. For example, the most likely item for one worker could be the least likely item for another. These differences could arise from varying cultural or regional biases, or alternate techniques for finding the data on the web. A mixture of multiple distributions over the same data results in a combined distribution that is "flatter" than its constituent parts, thereby becoming less skewed. In contrast, when the underlying data distribution is heavily skewed and shared amongst workers, the estimator will typically underestimate because there will not be a sufficient number of items representing the long tail of the distribution.

Figure 6(b) visualizes the impact of different data distributions combined with different worker skew on the Chao92 estimate by showing four combinations: the absence/presence of worker skew (WS) and the shared/different data distributions for workers (DD). For all cases, we use a power law data distribution in which the most likely item has probability p, the second-most likely has probability $p(1-p)$, etc.; we set $p = 0.03$. To simulate different data distributions, we randomly permute the original distribution for each worker.

The simulation shows that the worst scenario is characterized by a high worker skew and a single shared data distribution (WS=T and DD=F). With a shared skewed distribution, Chao92 will start out underestimating because all workers are answering with the same high-probability items. However, with high worker skew, the streaker(s) provide(s) many unique answers quickly causing many more unique items than encountered with sampling with replacement.

On the other hand, the best scenario is when there is no worker skew but there are different data distribution (WS=F and DD=T). By using different data distributions without overemphasizing a few workers, the overall sample looks more uniform, similar to Figure 6(a) with replacement, due to the flattening effect of DD on skewed data.

E. Worker Arrival

Finally, the estimate can be impacted by the arrival and departure of workers during the experiment. All workers do not necessarily provide answers during the lifetime of a query. Instead they come and go as they please. However, the estimator can be strongly impacted when streakers arrive who then suddenly dominate the total number of answers.

Figure 6(c) demonstrates the impact a single worker can have. It uses the same simulation setup as in Figure 6(b), but also simulates an additional single streaker starting at 200 HITs who continuously provides all 200 answers before anyone else has a chance to submit another answer. As the figure shows, it causes Chao92 to over-predict in all four cases. However, if workers use different data distributions the impact is not as severe. Again, this happens because DD makes the sample appear more uniformly distributed.

F. Discussion

Particular crowd behaviors are inherent in a marketplace like AMT. The order in which each worker provides his answers and how many he gives can depend on individual biases and preferences. The four elements of crowd behavior we outlined above (without-replacement sampling, worker skew, different distributions, and worker arrival) can each cause Chao92 to perform poorly. The most volatile of these behaviors is worker skew, particularly when the data distribution itself is skewed; a single overzealous worker could cause massive fluctuations in the estimate.

Our goal is to make Chao92 more fault-tolerant to the impact of such a streaker; we discuss our technique for a streaker-tolerant cardinality estimator next. In Section V, we revisit the scenario in which the estimator under-predicts due to one or more shared, highly skewed data distributions. In some cases, workers' answer sequences originate from a common list traversal. We develop a heuristic for detecting this behavior, which can be used to inform the decision to switch to an alternate data-gathering UI. Later on in Section VI, we describe a technique analyzing the cost versus benefit tradeoff of paying for more answers, helpful even when the cardinality estimate can only serve as a lower bound due to high data skew or inherent fuzziness in items' set membership.

IV. Streaker-Tolerant Completeness Estimator

Our goal is to provide the user with a progress estimate for an open-world query based on the answers that have been gathered so far. However, in the last section we demonstrated how having a crowd of humans enumerate a set creates a two-layer sampling process, and that the order in which items arrive depends heavily on different worker behaviors—which impacts the accuracy of the estimator.

In this section, we extend the Chao92 algorithm to make the estimator more robust against the impact of individual workers. We focus our effort mainly on reducing the impact of streakers and worker arrival, and exclude for now cases for which we can not make a good prediction, discussed in the following subsections in more detail. We first introduce the basic estimator model and Chao92 more formally before we present our extension that handles streaker impact. Finally, we evaluate our technique by first proposing a new metric that incorporates the notions of estimate stability and fast convergence to the true cardinality, then applying this metric to measure the effectiveness of our technique using various use cases in addition to the UN.

A. Basic Estimator Model and F-Statistic

Receiving answers from workers is analogous to drawing samples from some underlying distribution of unknown size N; each answer corresponds to one sample from the item distribution. We can rephrase the problem as a species estimation problem as follows: The set of HITs received from AMT is a sample of size n drawn from a population in which elements can be from N different classes, numbered $1 - N$ (N, unknown, is what we seek); c is the number of unique classes (species) seen in the sample. Let n_i be the number of elements in the sample that belong to class i, with $1 \leq i \leq N$. Of course some $n_i = 0$ because they have not been observed in the sample. Let p_i be the probability that an element from class i is selected by a worker, $\sum_{i=1}^{N} p_i = 1$; such a sample is often described as a multinomial sample [12].

One might try to estimate the underlying distribution $\{p_1, ..., p_N\}$ in order to predict the cardinality N. However, Burnham and Overton show in [17] that the aggregated "frequency of frequencies"-statistic (hereon f-statistic) is sufficient for estimating the number of unobserved species for non-parametric algorithms. The f-statistic captures the relative frequency of observed classes in the sample. For a population that can be partitioned into N classes (items), and given a sample of size n, let f_j be the number of classes that have exactly j members in the sample. Note f_1 represents the "singletons" and f_2 the "doubletons". The goal is to estimate the cardinality by predicting f_0, the number of unseen classes.

B. The Chao92 Estimator

Our technique is based on the Chao92 [14] estimator, which uses *sample coverage* to predict N. The sample coverage C is the sum of the probabilities p_i of the observed classes. However, since the underlying distribution $p_1 ... p_N$ is unknown, the Good-Turing estimator [19] using the f-statistic is used:

$$\hat{C} = 1 - f_1/n \tag{1}$$

Furthermore, the Chao92 estimator attempts to explicitly characterize and incorporate the skew of the underlying distribution using the *coefficient of variance* (CV), denoted γ, a metric that can be used to describe the variance in a probability distribution [14]; we can use the CV to compare the skew of different class distributions. The CV is defined as the standard deviation divided by the mean. Given the p_i's ($p_1 \cdots p_N$) that describe the probability of the ith class being selected, with mean $\bar{p} = \sum_i p_i/N = 1/N$, the CV is expressed as $\gamma = \left[\sum_i (p_i - \bar{p})^2/N\right]^{1/2} / \bar{p}$ [14]. A higher CV indicates higher variance amongst the p_i's, while a CV of 0 indicates that each item is equally likely.

The true CV cannot be calculated without knowledge of the p_i's, so Chao92 uses an estimate $\hat{\gamma}$ based on the f-statistic:

$$\hat{\gamma}^2 = \max\left\{\frac{\frac{c}{\hat{C}}\sum_i i(i-1)f_i}{n(n-1)} - 1\,,\,0\right\} \tag{2}$$

The final estimator is then defined as:

$$\hat{N}_{chao92} = \frac{c}{\hat{C}} + \frac{n(1-\hat{C})}{\hat{C}}\hat{\gamma}^2 \tag{3}$$

Note that if $\hat{\gamma}^2 = 0$ (i.e., indicating a uniform distribution), the estimator reduces to $c/\hat{C}$.

C. An Estimator for Crowdsourced Enumeration

The Chao92 estimator is heavily influenced by the presence of rare items in the sample; the coverage estimate $\hat{C}$ is based entirely on the percentage of singleton answers (f_1s). Recall from Section III the different crowd behaviors—many of them result in rapid arrival of unique answers. When unique items appear "too quickly", the estimator interprets this as a sign the complete set size is larger than it truly is. We develop an estimator based on Chao92 that ameliorates some of the overestimation issues caused by an overabundance of f_1 answers.

Most of the dramatic overestimation occurs in the presence of streakers, i.e., significant skew in the amount of answers provided by each worker. Notably, problems occur when one or a few workers contribute substantially more answers than others, possibly also drawing answers from a different data distribution. As other workers are not given the opportunity to provide answers that would subsequently increase the f_2s, f_3s, etc. in the sample, Chao92 predicts a full set cardinality that is too large. Thus our estimator is designed to identify any worker(s) who are outliers with respect to their contribution of unique answers in the sample (their f_1 answers).

The idea behind making the Chao92 estimator more resilient against streakers is to alter the f-statistic. The first step is to identify those workers who are "f_1 outliers". We define outlier in a traditional sense, two standard deviations outside the mean of all workers W. To avoid false negatives due to a true outlier's influence on the mean and standard deviation, both statistics are calculated without including the potential

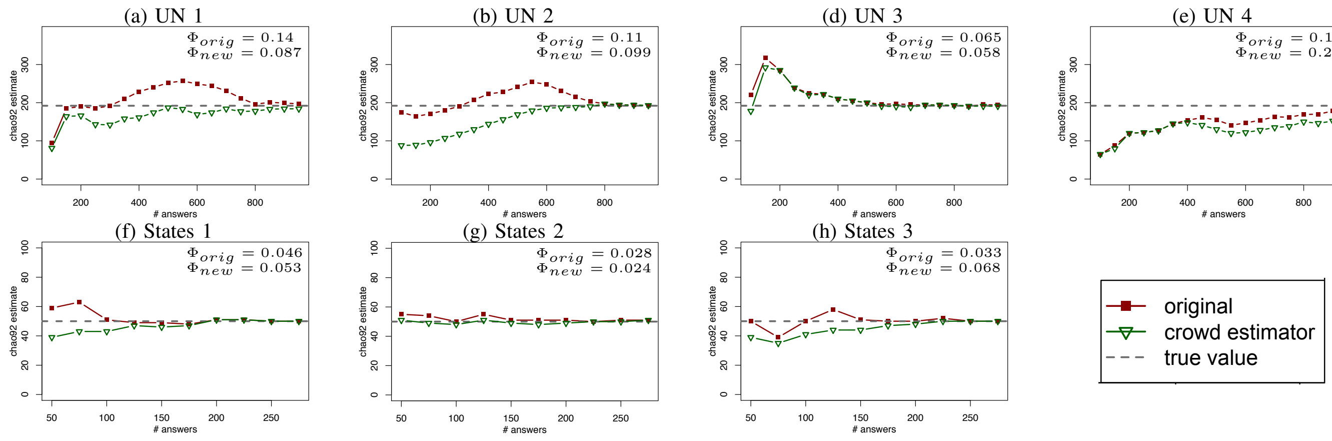


Fig. 7. Estimator results on representative UN country and US states experiments

outlier's f_1 count. The f_1 count of worker i is compared to the mean $\bar{x}_i$ and the sample standard deviation $\hat{\sigma_i}$:

$$\bar{x}_i = \sum_{\forall j, j \neq i} \frac{f_1(j)}{W-1} \quad \hat{\sigma_i} = \sqrt{\sum_{\forall j, j \neq i} \frac{(f_1(j) - \bar{x}_i)^2}{W-2}} \tag{4}$$

We create $\tilde{f_1}$ from the original f_1 by reducing each worker i's f_1-contribution to fall within $2\hat{\sigma_i} + \bar{x}_i$:

$$\tilde{f_1} = \sum_i min(f_1(i), 2\hat{\sigma_i} + \bar{x}_i) \tag{5}$$

The final estimator is similar to equation 3 except that it uses the $\tilde{f_1}$ statistic. For example, with a coefficient of variance $\hat{\gamma}^2 = 0$, it would simplify to:

$$\hat{N}_{crowd} = \frac{cn}{n - \sum_i min(f_1(i), 2\hat{\sigma_i} + \bar{x}_i)} \tag{6}$$

Although a small adjustment, $\hat{N}_{crowd}$ is more robust against the impact of streakers than the original Chao92, as we show in our evaluation next.

D. Experimental Results

We ran over 30,000 HITs on AMT for set enumeration tasks to evaluate our technique. Several `CROWD` tables we experimented with include small and large well-defined sets like NBA teams, US states, UN member countries, as well as sets that can truly leverage human perception and experience like indoor plants with low-light needs, restaurants in San Francisco serving scallops, slim-fit tuxedos, and ice cream flavors. Workers were paid \$0.01-\$0.05 to provide one item in the result set using the UI shown in Figure 3; they were allowed to complete multiple tasks if they wanted to submit more than one answer. In the remainder of this paper we focus on a subset of the experiments, some with known cardinality and fixed membership, US states (nine experiment runs) and UN member countries (five runs), as well as more open ended queries like plants, restaurants, tuxedos, and ice cream flavors (one run each).

1) Error Metric: Due to a lack of a good metric to evaluate estimators with respect to stability and convergence rate, we developed an error metric Φ that captures bias (absolute distance from the true value), as well as the estimator's time to convergence and stability. The idea is to weight the magnitude of the estimator's bias more as the size of the sample increases. Let N denote the known true value, and $\hat{N}_i$ denote the estimate after i samples. After n samples, Φ is defined as:

$$\Phi = \frac{\sum_{i=1}^{n} |\hat{N}_i - N| i}{\sum i} = \frac{2\sum_{i=1}^{n} |\hat{N}_i - N| i}{n(n+1)} \tag{7}$$

A lower Φ value means a smaller averaged bias and thus, a better estimate. The weighting renders a harsher penalty for incorrectness later on than in the beginning, in addition to penalizing an estimator that takes longer to reach the true value; this addresses the convergence rate criteria. The error metric also rewards estimators for staying near the true value.

2) Results: UN and States: We first illustrate how $\hat{N}_{crowd}$ behaves for a representative set of UN member countries and US states experiments; we elide the full set for space reasons. For both experiments the UI from Figure 3 was shown by CrowdDB to ask for an UN member country, respectively US state, on AMT for \$0.01 cents per task. Figures 7(a-h) show cardinality estimates as well as the Φ metric for the selected experiments. We observed that our estimate has an improvement over Chao92 for most UN experiments we performed as Figure 7(a) and (b) show. In UN 1 our estimates reduces the overestimation of Chao92 that occurred during the middle of the experiment. In the UN 2 experiment, one streaker dominated the total answer set at the beginning—a substantial outlier. Once his contribution was reduced dramatically, the remaining workers' answers had significant overlap because most were enumerating the list of nations alphabetically, resulting in a low cardinality because of the heavily skewed data distribution this scenario creates. Recall from the previous section that the expected behavior of the estimator in this case is to *under-predict*. In contrast, the third UN experiment run had several streakers at the beginning who each had very different data distributions (i.e., enumerating the list of nations from different alphabetical start points). While the heuristic helped level the f_1 contribution from these workers,

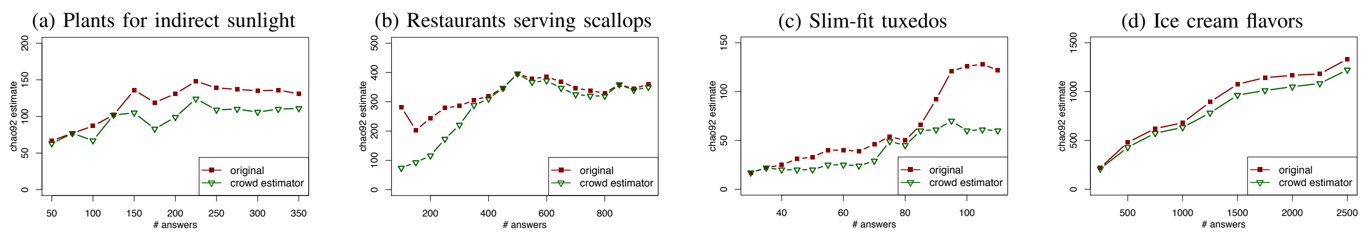


Fig. 8. Estimator results for the real use cases

overestimation still occurs due to the combined number of singleton answers from these workers. In a few cases, our estimator performs worse than Chao92, e.g., UN 4. Note that underestimation is expected when workers share a heavily skewed distribution; a streaker causing an estimate to be higher than it should results in a value closer to the true value.

The effect of our estimate compared to Chao92 is less significant in the States experiments, which exhibit less worker skew. Figure 7(f) and (g) show two US states experiments that have a moderate streaker problem and illustrate how our technique improves the prediction, whereas for a third state experiment shown in Figure 7(g), our estimator reduces the impact of streakers but takes longer to converge for similar reasons as in the UN 4 experiment.

3) Results: Real Use Cases: Our real use cases demonstrate several of the worker behaviors that we observed in the UN experiments as well; in particular, the presence of overzealous workers who contributed many more unique answers than others. Figures 8(a-c) show the original Chao92 and our estimates for the plants, restaurants, tuxedos, and ice cream flavors experiments. In all cases, our estimator successfully reduces the impact these streakers have on the prediction of complete set cardinality. Note that we cannot evaluate the error Φ for these experiments because the true cardinality is unknown. During the plant experiment, one worker from the beginning consistently contributed more unique answers than the other workers, e.g., "rabbit's foot"; many workers stuck to the well-known answers (e.g., snake plant, peace lily). In contrast, in the restaurant experiment a streaker contributed many f_1 answers at the beginning, but other workers eventually provided many of those same answers. The tuxedos experiment demonstrates how a streaker who arrives later in the experiment affects the estimate, causing a sharp increase in the Chao92 estimate which is ameliorated by $\hat{N}_{crowd}$.

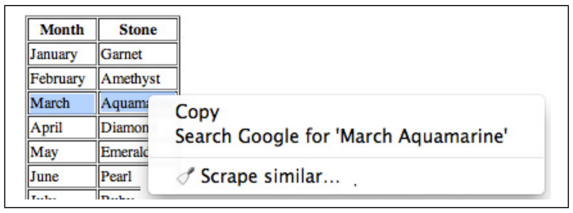
E. Discussion

In this section, we showed that our estimator successfully provides more accurate prediction for crowd-based set enumerations in the presence of overzealous workers (i.e., streakers). Our technique specifically tackles cardinality overestimation, which can be quite extreme and misleads the user into thinking he is lacking many more items in the set than he really is. It should be noted, however, that any heuristic, including ours, can only cope with a certain range of worker behavior that one could encounter when crowdsourcing a set. For example, if only one worker provides any answers, there is no information about the underlying data distribution for the estimator to take advantage of. On the other hand, if there are many workers producing few answers from a heavily skewed data distribution, an estimator is likely to underestimate because there will always be very few f_1 answers in the set. Most of the real experiments we ran on AMT did not fall into these extreme categories, and the heuristic is able to ameliorate the moderate impact of worker behavior on cardinality estimation.

V. List Walking

As described in Section III, when workers share the same or multiple heavily skewed data distribution, particularly if there is low worker skew, the estimator may under-predict the total set size. Such a heavily skewed distribution can occur if workers are traversing the same list for answers; we notice this behavior in some of our experiments. We refer to this effect as *list walking*. While unsurprising for the UN or States, list walking appears even in the ice cream flavor experiment.

Detecting list walking makes it possible to change the crowd-sourcing strategy to save money. For example, we are currently exploring how to use workers to scrape the data using a browser plugin shown in Figure 9, where workers mark the data to extract. In cases where one or two lists containing the full set exists, such as the UN countries, this switch could be helpful for getting them all. However, switching strategies for sets for which no single list exists (e.g., ice cream flavors) would not make sense. Thus the goal is to detect if list walking is particularly prominent in the set of workers' answers to inform the decision regarding data gathering UIs.



Month	Stone
January	Garnet
February	Amethyst
March	Aquama
April	Diamon
May	Emeral
June	Pearl

Fig. 9. Scraper Context Menu

In this section we devise a technique for detecting list walking based on the likelihood that multiple workers provide answers in the same exact order. We show that our technique can detect various degrees of list walking in our experiments.

A. Detecting lists

The goal of detecting list walking is to differentiate between samples drawn from a skewed item distribution and the existence of a list, which leads to a deterministic answer sequence. Simple approaches, such as looking for alphabetical order, finding sequences with high rank correlation or small edit-distance would either fail to detect non-alphabetical orders or disregard the case where workers return the same order simply by chance. In the rest of this section, we focus on a heuristic to determine the likelihood that a given number of workers w would respond with s answers in the exact same order.

List walking is similar to extreme skew in the item distribution; however even under the most skewed distribution, at some point (i.e., large w or large s), providing the exact same sequence of answers will be highly unlikely. Our heuristic determines the probability that multiple workers would give the same answer order if they were really sampling from the same item distribution. Once this probability drops below a particular threshold (we use 0.01), we conclude that list walking is likely to be present in the answers. We also consider cases of list walking with different offsets (i.e., both workers start from the fifth item on the list), but we do not consider approximate matches which may happen if a worker skips items on the list. Detecting list walking in those scenarios is considered as future work. Furthermore, approximate matches in answer order may make the sample more random and hence more desirable for estimation purposes.

1) Preliminary setup: binomial distribution: Let W be the total number of workers who have provided answer sequences of length s or more. Among these, let w be the number of workers who have the same sequence of answers with length s starting at the same offset o in common. We refer to this sequence as the *target sequence* α of length s, which itself is composed of the individual answers α_i at every position i starting with offset o ($\alpha = (\alpha_{o+1}, \ldots, \alpha_{o+s})$). If p_α is the probability of observing that sequence from some worker, we are interested in the probability that w out of W total workers would have that sequence. This probability can be expressed using the binomial distribution: W corresponds to the number of trials and w represents the number of successes, with probability mass function (PMF):

$$Pr(w; W, p_\alpha) = \binom{W}{w} p_\alpha^w (1 - p_\alpha)^{W-w} \tag{8}$$

Note that the combinatorial factor captures the likelihood of having w workers sharing the given sequence by chance just because there are many workers W. In our scenario, we do not necessarily care about the probability of exactly w workers providing the same sequence, but rather the probability of w *or more* workers with the same answer sequence:

$$Pr_{\geq}(w; W, p_\alpha) = 1 - \sum_{i=0}^{w-1} \binom{W}{i} p_\alpha^i (1 - p_\alpha)^{W-i} \tag{9}$$

The probability in equation 9 determines if the target sequence shared among w out of W workers is likely caused by list walking. We now discuss p_α, the probability of observing a particular target sequence α of length s.

2) Defining the probability of a target sequence: Not all workers use the same list or use the same order to walk through the list, so we want p_α to reflect the observed answer sequences from workers. We do this by estimating the probability $p_\alpha(i)$ of encountering answer α_i in the i^{th} position of the target sequence by the fraction of times this answer appears in the i^{th} position among all W answers. Let $r(i)$ be the number of times answer α_i appears in the i^{th} position among all the sequences W being compared, $p_\alpha(i)$ is defined as r_i/W. For example, if the target sequence α starting at offset o is "A,B,C" and the first answers for four workers are "A","A","A", and "B", respectively, r_{o+1}/W would be $3/4$. Now the probability of seeing α is a product of the probabilities of observing α_{o+1}, then α_{o+2}, etc.

$$p_\alpha = \prod_{i=o}^{o+s} \frac{r_i}{W} \tag{10}$$

Relying solely on the data in this manner could lead to false negatives in the extreme case where $w = W$, i.e., where all workers use the same target sequence. Note that in this case p_α attains the maximum possible value of 1. As a result, p_α will be greater than any threshold we pick. We need to incorporate *both* the true data via r_i/W as well as our most pessimistic belief of the underlying skew. As a pessimistic prior, we choose the highly skewed Grays self-similar distribution [20], often used for the 80/20 rule. Only if we find a sequence which can not be explained (with more than 1% chance) with the 80/20 distribution, we believe we have encountered list walking. Assuming a high skew distribution is conservative because it is more likely that workers will answer in the same order if they were truly sampling than with, say, a uniform distribution. The self-similar distribution with $h = 0.2$ is beneficial for our analysis because when sampling without replacement, the most likely item has 80% ($1 - h = 0.8$) chance of being selected and, once that item is selected and removed, the next most likely item has an 80% chance as well.

As a first step, we assume that the target sequence follows the self-similar distribution exactly by always choosing the most likely sequence. In this case α is simply a concatenation of the most likely answer, followed by the second most likely answer, and so on. Hence the likelihood of selecting this sequence under our prior belief is $(1 - h)^s$ and the likelihood that a set of w workers select this same sequence is:

$$(1 - h)^{sw} \tag{11}$$

Note that this probability does not calculate the probability of having *any given* sequence of length s shared among w workers; instead it represents the likelihood of having the most likely sequence in common. Incorporating the probability of all sequences of length s would be the sum of the probabilities of each sequence order, i.e., the most likely sequence plus the second most likely sequence, etc. However, we found that the terms after the most likely sequence contribute little and our implementation of that version had little effect on the results; thus do not consider it further.

To combine the distribution derived from data and our prior belief in the maximum skew, we use a smoothing factor β to shift the emphasis from the data to the distribution; higher values of β put more emphasis on the data. Using β to combine equation 10 with equation 11, we yield the probability of having the target sequence α (of length s) in common:

$$p_\alpha = \prod_{i=1}^{s} \left(\beta \frac{r_i}{W} + (1 - \beta)(1 - h) \right) \tag{12}$$

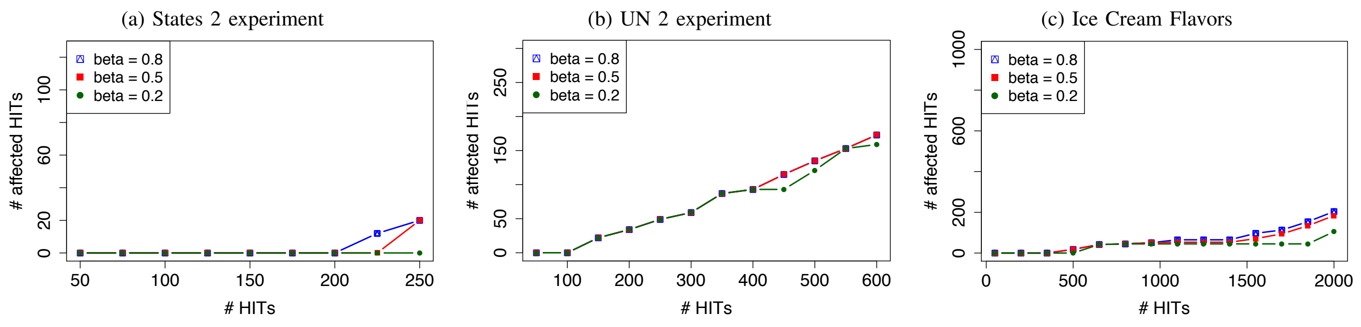


Fig. 10. HITs detected as list-walking for different experiments

If $\beta = 1$, p_α only incorporates the frequency information from the data, so if all workers are walking down the same list, then the probability in equation 12 would be 1 (thus not detecting the list use). Note also that when $\beta = 0$, p_α just uses the $80-20$ distribution and will reduce to $(1-h)^s$.We demonstrate the effect of different values of β next.

B. *Experimental Results*

To apply our heuristic to the AMT experiments, we prune the search space by using a window size s of at least 5 over the answers per worker. That is, for a sequence of answers of at least size s that have more than one worker in common, we compute the probability of that sequence using equation 8. If the probability falls below the threshold 0.01, we consider the sequence as being from a list. Our version of windowing ensures that we compare sequences that start at the same offset o across all workers, as equation 12 leverages the relative order that workers provide answers. A shingling approach to detect lists with different offsets across workers is beyond the scope of this paper. We check for list use over time (number of HITs) and quantify how many of the observed HITs were part of a list; this gives a sense of the impact of list use in the experiment. Due to space, we describe only a few results.

Figure 10 shows the number of affected HITs in one of the States experiments, one of the UN experiments, and for the ice cream flavors experiment. We use representative single runs as opposed to averages to better visualize the effect that a user of the systems would observe. The lines correspond to using equation 12 with different β values $0.2, 0.5, 0.8$. Lower β values detect fewer lists or it takes more HITs to detect lists.

The States experiments experienced little or no list walking. While there are definitely webpages that show the list of US states, perhaps it was not too much harder for workers to think of them on their own. All UN experiments exhibited some list use, with the list of course being the alphabetical list of countries that can be found online. However, we also notice that in one of the experiments a few workers went through the list in reverse alphabetical order. Interestingly, we also detect some list walking in the ice cream experiment, despite it being a personal question easily answerable without consulting a source online. After some searching for the original sources, we actually found a few lists used for ice cream flavors, like those from the "Penn State Creamery" and "Frederick's Ice Cream". Several lists were actually not alphabetical, including a list of the "15 most popular ice cream flavors" as well as forum thread on ChaCha.com discussing ice cream flavors.

Our results show that our heuristic is able to detect when multiple workers are consulting the same list and how severe list walking is. For example, it reports that for the UN 2 experiment around 20-25% of all HITs are impacted by list walking. Whereas for the ice cream flavors experiment less than 10% are impacted. So far we use the list walking detection to warn the user that the accuracy of the prediction might be impacted. In the future, we plan to automatically switch to alternative crowdsourcing strategies and ask the AMT workers to scrape the list with the UI shown in Figure 9.

VI. Cost vs. benefit: pay-as-you-go

While there are times list walking can be detected and the complete result set garnered by accessing the list directly, in many real use cases such lists are not available. Furthermore, the result set for some queries may have unbounded size, a highly skewed distribution and/or extreme worker behavior that make predicting its size nonsensical, as discussed in Section III. For these cases, it makes more sense to try to estimate the benefit of spending more money, i.e., predicting the shape of the SAC (e.g., Figure 1) in the near future. In this section we analyze *pay-as-you-go* techniques to predict this cost versus benefit tradeoff of getting more answers by expending additional effort.

A. *Estimating Benefit via Sample Coverage*

A query optimizer in an open-world system would want to estimate the benefit of increased crowdsourcing effort to consider the end user's quality goals. For the set enumeration query in CrowdDB, we are interested in how many more unique items would be acquired with m more HITs, given the current number of received answers. Again, we leverage techniques from the species estimation community, which developed techniques to evaluate the benefit of additional physical effort like setting more animal traps. To our knowledge, this is the first time that these techniques are applied in the context of database queries.

In [21], the authors derive an estimator (hereon *Shen*) for the expected number of species $\hat{N}_{Shen}$ that would be found in an increased sample of size m. The approach assumes we have an estimate of the number of unobserved elements $\hat{f}_0$ and that the unobserved elements have equal relative abundances.

Error: Average UN Experiments			
m	$n = 200$	$n = 500$	$n = 800$
10	1	0.6	0
50	5	1.6	1.6
100	8.4	3	3

Error: Average States Experiments			
m	$n = 50$	$n = 150$	$n = 200$
10	1	0.67	0.44
50	2.6	1.8	0.78
100	5.7	2	-

Error: Ice Cream Experiment			
m	$n = 1\text{K}$	$n = 1.5\text{K}$	$n = 2\text{K}$
10	4	0	0
50	9	1	0
100	13	1	3

Error: Plant Experiment			
m	$n = 100$	$n = 200$	$n = 300$
10	2	3	0
50	7	3	2
100	7	1	-

Error: Restaurant Experiment			
m	$n = 200$	$n = 400$	$n = 800$
10	2	0	3
50	11	5	13
100	18	3	-

Error: Tuxedo Experiment			
m	$n = 30$	$n = 50$	$n = 70$
10	3	0	1
50	8	10	-
100	-	-	-

Fig. 11. Pay-as-you-go cost-benefit predictions using *Shen*

However, this cardinality estimate $\hat{f}_0$ can incorporate a coefficient of variance estimate (equation 2) to account for skew, as shown by Chao92. An estimate of the unique elements found in an increased effort of size m is:

$$\hat{N}_{Shen} = \hat{f}_0 \left[1 - \left(1 - \frac{1 - \hat{C}}{\hat{f}_0} \right)^m \right] \quad (13)$$

We present results based on the Chao92 estimate of $\hat{f}_0$. Our estimator is designed to reduce the impact of streakers to yield a more accurate estimate of total set size, i.e., as $m \to \infty$, however disregarding the rapid arrival rate of new times can cause local predictions with $\hat{N}_{shen}$ to under-predict. Thus we use the original Chao92 estimate for the pay-as-you-go prediction.

Another technique [22] models the "expected mean" SAC with a binomial mixture model. It performs similarly to the coverage approach; we do not discuss it further.

B. Experimental Results

We evaluated the effectiveness of the $\hat{N}_{shen}$ estimator in determining how many more unique items would arrive if m additional answers were given by workers; this analysis would be done after having already received n answers (HITs). Accuracy is calculated as the absolute value of the error (bias); the absolute value allows for averaging the errors. The tables in Figure 11 contain the errors for various queries. For example, in the UN experiment after $n = 500$ the average error for the next $m = 50$ HITs is 1.6 (i.e., on average the prediction is off by 1.6). For some cells, we were not able to evaluate the prediction as we did not receive enough answers for at least one of the experiments. These cases are marked with an dash.

For all experiments, predictions for small m are easier since only the near future is considered, thus they tend to be more accurate. The larger the m, the further the prediction has to reach and thus the more error-prone the result, particularly if m exceeds the current HITs size n [21]). The pay-as-you-go results are also aligned with the intuition the SAC provides: at the beginning when there are few worker answers, it is fairly inexpensive to acquire new unique items. Towards the end, more unique items are hard to come by.

Worker behavior also has an influence on the pay-as-you-go predictions. The Shen estimator tends to under-predict before the accumulation curve plateaus, as the curve is steeper than expected. This happens because workers sample without replacement—unique answers appear more quickly than they would from a with-replacement sample. While minimizing all error is ideal, under-prediction is not catastrophic since the user will end up getting more bang for his buck than anticipated. There is also potential to use knowledge of worker skew and particularly the presence of streakers to inform the user when an under-prediction is likely. Thus $\hat{N}_{shen}$ provides a reasonable mechanism for the user to analyze the cost-benefit tradeoff of acquiring more answers in the set.

VII. Related Work

In this paper we focused on estimating progress towards completion of a query result set, an aspect of query quality. To our knowledge, quality of an open-ended question posed to the crowd has not been directly addressed in literature. In contrast, techniques have been proposed for quality control for individual set elements [5], [8].

Our estimation techniques build on top of existing work on species or class estimation [12], [9], [13]. These techniques have also been used and extended in database literature for distinct value estimation [11], [23]. For example, a hybrid approach is proposed in [11], choosing between the Shlosser estimator [24] and a version of the Jackknife estimator [17] the authors modified to suit a finite population (i.e., known table size). This approach is further improved and extended by [23] to derive a lower bound on error. Unfortunately, both the error bounds and developed estimators for distinct values in a table explicitly incorporate knowledge of the full table size, possible only in the closed world. Furthermore, none of the techniques consider the sampling scenario of crowdsourced queries and are therefore not applicable as discussed in Section III.

Species estimation techniques were also explored for search and meta-search engines. For example, in [25] the authors develop an algorithm to estimate the size of any set of documents defined by certain conditions based on previously executed queries. Whereas [26] describes an algorithm to estimate the corpus size for a meta-search engine in order to better direct queries to search engines. Similar techniques are also used to measure the quality of search engines [27]. All techniques differ from those described in this paper, as they do not consider the specific worker behavior and assume sampling with replacement.

Recent work also tries to explore species estimation techniques for the deep web [28], [29]. Again, the proposed techniques have strong assumptions regarding the sample which do not hold in the crowd setting. Some of these assumptions might not even hold in the context of the deep web. We believe that our techniques, which are more robust against biased samples, are applicable in the context of deep web search/data integration and consider it future work.

Although this work was done as part of CrowdDB [2], it could be applied to other hybrid human-machine database systems, such as Qurk [3] or sCOOP [30]. Both systems allow for acquiring sets from the crowd but do not yet provide any quality control mechanisms for it.

There is an array of literature on crowdsourcing in general, addressing issues from improving and controlling latency [31], [6] to correcting the impact of worker capabilities [32]. This work is orthogonal to estimating the set quality.

VIII. Future work and Conclusion

People are particularly well-suited for gathering new information because they have access to both real-life experience and online sources of information. Incorporating crowdsourced information into a database, however, raises questions regarding the meaning of query results without the closed-world assumption – how does one even reason about a simple `SELECT *` query? We argue that progress estimation allows the user to make sense of query results in the open world. We develop techniques for analyzing progress via result set cardinality estimation that consider crowd behaviors, based on species estimation algorithms.

Many future directions exist, ranging from different user interfaces for soliciting worker input to incorporating the above techniques into a query optimizer. We have done initial explorations into a "negative suggest" UI that only allows workers to enter answers no one has yet provided. A hybrid approach with this interface and the current interface could be used to expand an existing set and/or target rare items. In this paper, we assume that workers do not provide incorrect answers, as a variety of quality control solutions for single answers are proposed already in the literature. However, fuzzy set membership (e.g., is Pizza or Basil a valid ice-cream flavor) imposes interesting new challenges on quality control for sets. Techniques developed for crowdsourced queries in databases are readily applicable to deep web queries. Human perception is advantageous in answering these queries, and the question of query completeness appears in this context as well.

Using statistical techniques we enable users to reason about query progress and cost-benefit trade-offs in the open world.

Acknowledgment

This work was inspired by the CrowdDB project, and we would like to thank our CrowdDB collaborators Donald Kossmann, Sukriti Ramesh, and Reynold Xin.

This research is supported in part by a National Science Foundation graduate fellowship, in part by NSF CISE Expeditions award CCF-1139158, and gifts from Amazon Web Services, Google, SAP, Blue Goji, Cisco, Cloudera, Ericsson, General Electric, Hewlett Packard, Huawei, Intel, Microsoft, NetApp, Oracle, Quanta, Splunk, VMware and by DARPA (contract #FA8650-11-C-7136).

References

[1] J. C. R. Licklider, "Man-Computer Symbiosis," *IRE Trans. of Human Factors in Electronics*, vol. 1, 1960.

[2] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: Answering Queries with Crowdsourcing," in *Proc. of SIGMOD*, 2011.

[3] A. Marcus, E. Wu, S. Madden, and R. Miller, "Crowdsourced Databases: Query Processing with People," in *Proc. of CIDR*, 2011.

[4] A. Parameswaran and N. Polyzotis, "Answering Queries using Humans, Algorithms and Databases," in *Proc. of CIDR*, 2011.

[5] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on Amazon Mechanical Turk," in *Proc. of HCOMP*, 2010.

[6] D. W. Barowy, E. D. Berger, and A. McGregor, "AUTOMAN: A Platform for Integrating Human-Based and Digital Computation," University of Massachusetts, Tech. Rep., 2011.

[7] K.-T. Chen *et al.*, "A crowdsourceable QoE evaluation framework for multimedia content," in *ACM Multimedia*, 2009.

[8] A. Doan, M. J. Franklin, D. Kossmann, and T. Kraska, "Crowdsourcing Applications and Platforms: A Data Management Perspective," *PVLDB*, vol. 4, no. 12, 2011.

[9] R. K. Colwell and J. A. Coddington, "Estimating Terrestrial Biodiversity through Extrapolation," *Philosophical Trans.: Biological Sciences*, vol. 345, no. 1311, 1994.

[10] A. Feng *et al.*, "CrowdDB: Query Processing with the VLDB Crowd," *PVLDB*, vol. 4, no. 12, 2011.

[11] P. J. Haas *et al.*, "Sampling-based estimation of the number of distinct values of an attribute," in *Proc. of VLDB*, 1995.

[12] J. Bunge and M. Fitzpatrick, "Estimating the Number of Species: A Review," *Journal of the American Statistical Association*, vol. 88, no. 421, 1993.

[13] A. Chao, "Species Richness Estimation," http://viceroy.eeb.uconn.edu/EstimateSPages/ EstSUsers-Guide/References/Chao2005.pdf, 2005.

[14] A. Chao and S. Lee, "Estimating the Number of Classes via Sample Coverage," *Journal of the American Statistical Association*, vol. 87, no. 417, pp. 210–217, 1992.

[15] J. Bunge, M. Fitzpatrick, and J. Handley, "Comparison of three estimators of the number of species," *Journal of Applied Statistics*, vol. 22, no. 1, pp. 45–59, 1995.

[16] A. Chao, "Nonparametric Estimation of the Number of Classes in a Population," *SJS*, vol. 11, no. 4, 1984.

[17] K. P. Burnham and W. S. Overton, "Estimation of the Size of a Closed Population when Capture Probabilities vary Among Animals," *Biometrika*, vol. 65, no. 3, 1978.

[18] J. Heer and M. Bostock, "Crowdsourcing graphical perception: using mechanical turk to assess visualization design," in *Proc. of CHI*, 2010.

[19] I. J. Good, "The Population Frequencies of Species and the Estimation of Population Parameters," *Biometrika*, vol. 40, no. 3/4, 1953.

[20] J. Gray *et al.*, "Quickly generating billion-record synthetic databases," in *Proc. of SIGMOD*, 1994.

[21] T. Shen, A. Chao, and C. Lin, "Predicting the Number of New Species in Further Taxonomic Sampling," *Ecology*, vol. 84, no. 3, 2003.

[22] R. K. Colwell, C. X. Mao, and J. Chang, "Interpolating, Extrapolating, and Comparing Incidence-Based Species Accumulation Curves," *Ecology*, vol. 85, no. 10, 2004.

[23] M. Charikar *et al.*, "Towards Estimation Error Guarantees for Distinct Values," in *Proc. of PODS*, 2000.

[24] A. Shlosser, "On Estimation of the size of the dictionary of a long text on the basis of a sample," *Engrg. Cybernetics*, vol. 19, 1981.

[25] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu, "Estimating corpus size via queries," in *Proc. of CIKM*, 2006.

[26] K.-L. Liu, C. Yu, and W. Meng, "Discovering the representative of a search engine," in *Proc. of CIKM*, 2002.

[27] Z. Bar-Yossef and M. Gurevich, "Efficient search engine measurements," *ACM Trans. Web*, vol. 5, no. 4, pp. 18:1–18:48, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2019643.2019645

[28] J. Lu and D. Li, "Estimating deep web data source size by capture—recapture method," *Inf. Retr.*, vol. 13, no. 1, pp. 70–95, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10791-009-9107-y

[29] J. Liang, "Estimation Methods for the Size of Deep Web Textural Data Source: A Survey," cs.uwindsor.ca/richard/cs510/survey_jie_liang.pdf, 2008.

[30] A. Parameswaran *et al.*, "Deco: Declarative Crowdsourcing," Stanford University, Tech. Rep., 2011.

[31] M. S. Bernstein *et al.*, "Crowds in Two Seconds: Enabling Realtime Crowd-Powered Interfaces," in *Proc. of UIST*, 2011.

[32] P. Welinder *et al.*, "The Multidimensional Wisdom of Crowds," in *Proc. of NIPS*, 2010.