Answering Enumeration Queries with the Crowd

By Beth Trushkowsky, Tim Kraska, Michael J. Franklin, and Purnamrita Sarkar

Abstract

Hybrid human/computer database systems promise to greatly expand the usefulness of query processing by incorporating the crowd. Such systems raise many implementation questions. Perhaps the most fundamental issue 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 nonuniformities 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.

1. INTRODUCTION

A number of recent projects have demonstrated that leveraging crowdsourcing can greatly extend the usefulness of a query processing system, for example, CrowdDB,¹¹ Quirk,¹⁹ and sCOOP.²⁰ In these systems, human workers can be called upon to perform query operations such as subjective comparisons, fuzzy matching for predicates and joins, entity resolution, etc.

Of course, many challenges arise when adding people to query processing due to the peculiarities in latency, cost, quality, and predictability of human workers. Researchers have addressed many of these concerns, but we observe that adding the crowd to a database query processor raises issues of an even more fundamental semantic nature. Relational query languages are based on the *closed-world assumption*, in which the database is considered to be complete at the time a query is posed. That is, it contains all data needed to answer the query. When the crowd can be enlisted to add new data during query processing, this assumption is violated, calling into question the meaning of even simple queries.

1.1. Can you get it all? Enumeration queries

In this paper, we consider one of the most basic operations in a Relational Database Management System (RDBMS), namely, scanning a single database table with filtering constraints, or *predicates*; this query enumerates a particular set of items of interest to the user. Consider, for example, a SQL query to list all house plants that can tolerate low-light environments: `SELECT DISTINCT name FROM Plants WHERE light-needs = 'low'`. With a traditional RDBMS and a

given database state 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. 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.

In contrast, in a crowdsourced system such as 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. These types of queries are the main use cases for crowd-enabled database systems, as each is labor-intensive for the user issuing the query to perform, but not executed frequently enough to justify the development, tuning, and use of 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 tools that enable users to reason about time/cost and completeness trade-offs, and that can be used to drive query execution and crowdsourcing strategies.

1.2. 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 DISTINCT *`” 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 such as Amazon’s Mechanical Turk (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.

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.

The original version of this paper is entitled “Crowdsourced Enumeration Queries” and was published in ICDE, 2013, IEEE.²²

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 biostatistics, where this type of figure is known as the *Species Accumulation Curve* (SAC).⁹ Imagine you are 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, you can infer the true, albeit hidden, number of species. We can apply this reasoning to enumeration queries in a crowdsourced query processor.

1.3. 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 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 result set size estimation in this environment.

We also describe methods to leverage these techniques to help users make intelligent trade-offs between time/cost and completeness. The usefulness of these techniques extends beyond crowdsourced databases, for example, to estimate the completeness of deep-web queries.

To summarize, we make the following contributions:

- We formalize the process of crowdsourced enumeration and describe how it violates fundamental statistical assumptions of existing species estimation techniques.
- We develop a technique to estimate result set size, or *cardinality*, and query progress in the presence of crowd-specific behaviors.

Figure 1. U.S. States: unique versus total number of answers.

- We devise a technique to determine if data scraping could be applied, as well as describe a pay-as-you-go approach to allow informed decisions about the cost/completeness tradeoff.
- We examine the effectiveness of our techniques via experiments using AMT.

2. BACKGROUND: CROWDDB

Crowddb is a hybrid human-machine database system that uses human input to process queries. Crowddb supports different crowdsourcing platforms; we focus on AMT in this paper, the leading platform for the so-called microtasks. Microtasks usually do not require any special training and do not take more than a few minutes to complete. AMT provides a marketplace for requesters to post microtasks and workers to search for and work on these tasks for a small reward, typically a few cents.

Crowddb incorporates traditional query compilation, optimization, and execution components, which are extended to cope with human-generated input. 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. The UI Manager is responsible for generating the task interface for the microtasks that crowd workers interact with. Crowddb automatically generates UIs as HTML forms based on CROWD annotations. Note that in contrast to traditional database system front end UIs that enable end users to construct and submit query requests, these UIs are part of the database system back end and are presented to crowd workers who are assisting with the actual query execution. Figure 2 shows an example HTML UI that would be presented to a worker for the following crowd table definition:

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

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 a majority vote across crowd workers 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 Franklin et al.¹¹ This

Figure 2. Ice cream flavors task UI on AMT.

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:

paper focuses on the progress component that allows the user to continuously reason about query completeness and cost.

3. PROGRESS ESTIMATION

To evaluate progress as answers are arriving, the system needs an estimate of the result set's size, also called *cardinality*, in order to calculate the percentage complete. Species estimation algorithms tackle a similar goal: an estimate of the number of distinct species is determined using observations of species in the locale of interest. In this section, we describe observations of how the crowd answers enumeration queries and the challenges of estimation for crowdsourced queries. We present a model for crowdsourced enumerations and list the requirements for human-tolerant estimators.

3.1. The problem with existing estimators

Various techniques have been devised to estimate the number of species^{3, 6} and to estimate the number of distinct values in a database table.¹⁴ 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 true and unknown number of distinct values is estimated. The techniques differ most notably in their assumptions, in particular 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 that allow crowdsourced enumerations, records can be acquired on-demand, thus the table size is potentially infinite. We focus on estimators suitable for the open world as they allow for an infinite population.

To gain an understanding of the crowd's ability to answer enumeration queries and the impact of crowd behaviors on existing estimation techniques, we crowdsourced the elements of sets for which the true cardinality is known. We use the open-world-safe estimator "Chao92"⁷ as it is widely used in the species estimation literature.⁴ Figure 3 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

Figure 3. Chao92 cardinality estimate evaluated for increasing number of samples, or answers from the crowd, for the United Nations use case.

and compares it to the expected behavior using simulation with the empirical data distribution (DD) derived from all runs of the UN experiment. 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 we describe next.

Note in Figure 3 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 3). As it turns out, the way in which crowd workers each provide their answers deeply impacts the behavior of an estimation algorithm. For example, some workers enumerated the UN countries by traversing an alphabetical list. However, some workers began 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. In general, people may use different internal biases or techniques for finding items in the set (we discuss full list traversals in Trushkowsky et al.²²). Furthermore, individual workers complete different amounts of work and arrive/depart from the experiment at different points in time.

3.2. A model for human enumerations

Species estimation algorithms assume a with-replacement sample from some unknown distribution describing item likelihoods (visualized in Figure 4a). The order in which elements of the sample arrive is irrelevant in this context.

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

This process of sampling significantly differs from what traditional estimators assume, and it can be represented as

Figure 4. Comparison of the sampling processes, (a) assumed by traditional algorithms and (b) observed in crowd-based enumerations.

a two-layer sampling process as shown in Figure 4b. The bottom layer consists of many sampling processes, each corresponding to an individual worker, that sample from some DD *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 among workers who are each sampling a DD without replacement.

Next we discuss the impact of the parameterization of the two-layer sampling process (e.g., the number of worker processes, different underlying distributions, etc.) on estimation.

3.3. Sampling without replacement and worker skew

Individuals are sampling without replacement from some underlying distribution that describes the likelihood of selecting each item. 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, however, is that the estimator receives less information about the relative frequency of items, and thus the skew, of the underlying DD. This can cause the estimator to over-predict due to the more rapid arrival of unseen items than would occur in a with-replacement sample.

Over-prediction also results when some workers complete many more HITs than others; workers who do significantly more work have been called “streakers.”¹⁵ In the two-layer sampling process, worker skew (WS) dictates which worker supplies the next answer—skew in the ρ process; streakers are chosen with higher frequency. High WS can cause the arrival rate of unique answers to be more rapid than that caused by sampling without replacement alone, causing the estimator to over-predict.

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 workers each provide one answer using the same underlying distribution, the resulting sample would correspond to the sampling with replacement scenario (Figure 4a). The latter is the reason why it is still possible to make

estimations despite human-generated enumerations.

To illustrate the impact on the Chao92 estimator of the number of workers participating in a crowd-based enumeration, we simulated different numbers of workers sampling from a uniform distribution over 200 items, averaging over 100 runs. Figure 5a depicts the values of the Chao92 estimator calculated for increasing numbers of samples for three scenarios: a with-replacement sample (equivalent to an infinite number of workers), and three or five workers each sampling without replacement from the item distribution. As expected, the with-replacement sample overestimates slightly because of the uniform DD, but quickly approaches the true value of 200. The without-replacement samples, having fewer workers, overestimate even more and remain in that state for longer.

3.4. Different and skewed data distributions

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

Figure 5b shows the impact of different DDs combined with different WS on the Chao92 estimate. It shows four combinations: the absence/presence of WS and shared/different DDs for workers. For all cases, we use a power law DD in which the most likely item has probability p , the second most likely has probability $p(1-p)$, and the k th most likely item has probability $p(1-p)^{k-1}$, etc.; we set $p = 0.03$. To simulate different DDs, we randomly permute the original distribution for each worker.

The simulation shows that the worst scenario is characterized by a high WS and a single shared DD (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. With high WS, 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 WS but there are different DD (WS = F and DD = T). By using

Figure 5. Cardinality estimation simulations illustrating the impact of worker behaviors, (a) With vs. without replacement, (b) Forms of skew, and (c) Impact of streaker.

different DDs without overemphasizing a few workers, the overall sample looks more uniform, similar to Figure 5a with replacement, due to the flattening effect of DD on skewed data.

3.5. Worker arrival

Finally, the estimate can be impacted by the arrival and departure of workers during the experiment. Not all workers provide answers during the lifetime of a query. However, the estimator can be strongly impacted when streakers arrive who then suddenly dominate the total number of answers.

Figure 5c demonstrates the impact a single worker can have. It uses the same simulation setup as in Figure 5b, but also includes an additional single stalker 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 DDs the impact is not as severe. Again, this happens because DD makes the sample appear more uniformly distributed.

3.6. Discussion

Some of the behaviors that workers exhibit as they respond to enumeration queries are inherent in a marketplace such as AMT, such as how many tasks an individual worker chooses to complete. The order in which each worker provides his/her answers and how many he/she gives can depend on individual biases and preferences. The four elements of crowd behavior we outlined above (without-replacement sampling, WS, different distributions, and worker arrival) can each cause Chao92 to perform poorly. The most volatile of these behaviors is WS, particularly when the DD itself is skewed; a single overzealous worker could cause massive fluctuations in the estimate. Overestimation in particular is problematic because it may lead to unnecessary crowdsourcing costs in an attempt to enumerate more items of the set that do not actually exist. However, we do not want to prohibit individual workers, especially highly productive ones, from submitting responses, as doing so would slow or limit the progress of the query. Thus we want to make Chao92 more tolerant to the impact of such a stalker while still allowing the stalker to submit answers; we discuss our technique for a stalker-tolerant cardinality estimator next.

4. STALKER-TOLERANT ESTIMATOR

Our goal is to provide a progress estimate for an open-world query based on the answers that have been gathered so far. In this section, we extend the Chao92 estimator to be more robust to the impact of individual workers, focusing our effort on reducing the impact of stalkers and worker arrival. We first introduce the basic estimator model and Chao92 more formally before we present our extension to handle stalkers. We evaluate our technique by first proposing a new metric that incorporates notions of estimate stability and fast convergence to the true cardinality, then applying this metric to measure our technique's effectiveness.

4.1. 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, $\sum_{i=1}^N p_i = 1$.

Burnham and Overton⁵ show that the aggregated “frequency of frequencies”-statistic (hereon f -statistic) is sufficient for estimating the number of unobserved species for nonparametric algorithms. The f -statistic captures the relative frequency of observed classes in the sample. Let f_j be the number of classes that have exactly j members in the sample. The goal is to estimate the cardinality by predicting f_0 , the number of unseen classes.

4.2. The Chao92 estimator

The Chao92⁷ estimator uses *sample coverage* to predict N . The sample coverage C is the sum of the probabilities p_i of the observed classes. Since the underlying distribution $p_1 \dots p_N$ is unknown, the Good-Turing estimator¹² using the f -statistic is used:

$$\hat{C} = 1 - \frac{f_1}{n} \quad (1)$$

The Chao92 estimator attempts to explicitly characterize and incorporate the skew of the underlying distribution using a *coefficient of variance* (CV) γ , a metric that is used to describe the variance in a probability distribution⁷; 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 ($p_1 \dots p_N$) that describe the probability of the i th class being selected, with mean $\bar{p} = \sum_i p_i / N = 1/N$, the CV is expressed as $\gamma = [\sum_i (p_i - \bar{p})^2 / N]^{1/2} / \bar{p}$.⁷ Higher CV means higher variance in the p_i 's; a CV of 0 means each item is equally likely.

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

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

The final estimator is then defined as:

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

4.3. 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_1 s). Recall from Section 3 the discussion of different crowd behaviors—many of these result in rapid arrival of previously unseen answers. When these new f_1 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 due to streakers, that is, significant skew in the number 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 DD. Since other workers are not given the opportunity to provide answers that would subsequently increase the f_2 s, f_3 s, etc. in the sample, Chao92 predicts a total 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 traditionally, namely, 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 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 $\hat{\sigma}_i$:

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

We create $\tilde{f}_1$ from the original f_1 by reducing each worker i ’s f_1 -contribution to fall within $2\hat{\sigma}_i + \bar{x}_i$:

$$\tilde{f}_1 = \sum_i \min(f_1(i), 2\hat{\sigma}_i + \bar{x}_i) \quad (5)$$

The final estimator is similar to Equation (3) except that it uses the $\tilde{f}_1$ statistic. For example, with a CV $\hat{\gamma}^2 = 0$, it would simplify to:

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

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

4.4. Experimental results

We ran more than 30,000 HITs on AMT for enumeration tasks. The 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 such as 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 a UI similar to that in Figure 2; 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, two with known cardinality and fixed membership, US states (nine experiment runs) and UN member countries (five runs), as well as more open-ended queries (one run each).

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:

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

Lower Φ means a smaller averaged bias and thus a better estimate. The weighting gives 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, addressing the convergence rate criteria. The metric also rewards estimators for staying near the true value.

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. As discussed in Section 3, we do not average the results across experiment runs because each run may have different data and WSSs; averaging could disguise the impacts of worker behaviors on the estimators.

Figure 6a–g show cardinality estimates as well as the Φ metric for these experiments. Each graph shows the Chao92 algorithm estimates (labeled “original”) and the value of the error metric calculated for those estimates (Φ_{orig}), as well as the estimates and error (Φ_{new}) for the streaker-tolerant estimator (labeled “crowd estimator”). We observed that our estimate has an improvement over Chao92 for most UN experiments.

In the experiment run labeled UN 1, our estimates avoids 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/her 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 DD this scenario creates. Recall from the previous section that the expected behavior of the estimator in this case is to *underpredict*. In contrast, the third UN experiment run had several streakers at the beginning who each had very different DDs (i.e., enumerating the list of nations from different alphabetical start points). While the heuristic helped level the f_1 contribution from these workers, overestimation still occurs due to the combined number of singleton answers from them. In a few cases, our estimator performs worse than Chao92, for example, experimental run UN 4. Underestimation is expected when workers share a highly skewed distribution; a streaker causing an estimate to be higher than it should incidentally yields a value closer to the true value.

The effect of our estimate compared to Chao92 is less significant in the States experiments, which had less WS. Figure 6e and f show two US states experiments that have moderate streaker issues, helped by $\hat{N}_{crowd}$. In a third state experiment (Figure 6g), our estimator reduces the streakers’ impact but takes longer to converge for similar reasons as in run 4 of the UN experiment.

Results: Open-ended use cases. The UN countries and U.S. states use cases are both sets for which the true

cardinality, as well as the sets' contents, is known; use cases with known cardinality allow us to evaluate the accuracy of the estimation algorithms. Here we look at use cases that are “open-ended,” that is, the set contents and cardinality are not known. For these results, we can only compare estimation algorithms. The open-ended use cases demonstrate several of the worker behaviors that we observed in the UN experiments; in particular, the presence streakers. Figure 7a–d show Chao92 and our $\hat{N}_{crowd}$ 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 (Figure 7a), one worker from the beginning consistently contributed more unique answers than the other workers, for example, a less well-known plant called “rabbit’s foot”; many workers stuck to well-known answers (e.g., snake plant, peace lily). In contrast, in the restaurant experiment (Figure 7b) a streaker contributed many f_1 answers at the beginning, but other workers eventually provided many of those same answers. The tuxedos experiment (Figure 7c) shows the impact of a streaker arriving later in the experiment, causing a sharp increase in the Chao92 estimate which is helped by $\hat{N}_{crowd}$.

4.5. Discussion

We showed that our estimator successfully provides more accurate prediction for crowd-based enumerations in the presence of overzealous workers. Our technique specifically tackles cardinality overestimation, which can cause the user issuing the query to think the result set is less complete than it really is. However, any estimator can only cope with a certain range of worker behavior. The extreme cases in which only one worker provides answers, or if workers share a heavily skewed distribution, will prove difficult for an estimator. Most of the experiments we ran did not have these issues, and the heuristic is able to ameliorate the impact of worker behavior on estimation.

5. LIST WALKING

When workers share the same or multiple heavily skewed DD, 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 refer to this effect as *list walking*. Detecting list walking makes it possible to change the crowdsourcing strategy to save money. 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 which no single list exists would not make sense.

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

Figure 6. Estimator results on representative UN country and U.S. states experiments, (a) UN 1, (b) UN 2, (c) UN 3, (d) UN 4, (e) States 1, (f) States 2, and (g) States 3.

Figure 7. Estimator results for the open-ended use cases, (a) Plants for indirect sunlight, (b) Restaurants serving scallops, (c) Slim-fit tuxedos, and (d) Ice cream flavors.

sequence. In this section, we develop a heuristic to determine the probability that a given number of workers w would respond with s answers in the exact same order. If this probability is below a threshold (we use 0.01), we conclude that list walking is likely to be present.

5.1. 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 ($\alpha = (\alpha_{o+1}, \dots, \alpha_{o+s})$). 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. Furthermore, 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. This probability can be expressed using the binomial distribution: W corresponds to the number of trials and w represents the number of successes:

$$Pr \geq (w; W, p_\alpha) = 1 - \sum_{i=0}^{w-1} \binom{W}{i} p_\alpha^i (1-p_\alpha)^{W-i} \quad (8)$$

The probability in Equation (8) 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 .

5.2. 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.

$$p_\alpha = \prod_{i=0}^{o+s} \frac{r_i}{W} \quad (9)$$

Relying solely on the data in this manner could lead to false negatives in the extreme case where $w = W$, that is, where all workers use the same target sequence. Note that in this case p_α attains the maximum possible value of 1. We need to incorporate *both* the true data via r_i/W as well as a pessimistic belief of the underlying skew. As a pessimistic prior, we choose the highly skewed Gray's self-similar distribution,¹³ 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. 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. 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 sequence is:

$$(1-h)^{sw} \quad (10)$$

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 (9) with Equation (10), we yield the probability of having the target sequence α (of length s) in common:

$$p_\alpha = \prod_{i=1}^s \left(\beta \frac{r_i}{W} + (1-\beta)(1-h) \right) \quad (11)$$

5.3. Experimental results

We apply our heuristic to the AMT experiments, looking at target sequences α of at least length $s = 5$. 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). A sequence is considered a list if the probability falls below the threshold 0.01. 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 results for one run of the UN experiment.

Figure 8 shows the number of affected HITs in one of the UN experiments. The lines correspond to using Equation (11) with different β values 0.2, 0.5, 0.8. Lower β values detect fewer lists (or it takes more HITs to detect lists).

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. The States experiments had little or no list walking. Despite webpages that show the list of US states, we posit

Figure 8. List walking in UN experiment run 2.

that workers did not find thinking of the states' names too difficult to warrant consulting the web.

The open-ended use case also contained little or no list walking. In the future, we plan to detect list walking and switch to scraping information from the web.

6. COST VERSUS BENEFIT: PAY-AS-YOU-GO

For sets with finite membership, it makes sense to estimate the set size and employ the crowd to provide the complete set. However, 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 3. For these cases, it makes more sense to try to estimate the benefit of spending more money, that is, predicting the shape of the SAC (e.g., Figure 1) in the near future.

Again, we leverage species estimation algorithms to create *pay-as-you-go* techniques to predict this cost versus benefit tradeoff of getting more answers by expending additional effort. We evaluated the effectiveness of an estimator developed by Shen et al.²¹ in determining how many unseen items would arrive with m additional answers from workers; this analysis would be done after having already received n answers (HITs). In general, we found that predictions for small m are more accurate since only the near future is considered. 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 A full description of the pay-as-you-go approach and experimental results is in Trushkowsky et al.²²

7. RELATED WORK

In this paper, we focused on estimating progress toward completion of a query result set, an aspect of query quality. Techniques have been proposed for quality control that can be used for verification of individual set elements.^{10, 16} This work was done as part of CrowdDB,¹¹ but it could be applied to other hybrid database systems that use the crowd to enumerate sets.

Our estimation techniques build on top of existing work on species estimation.^{3, 6, 9} These techniques have also been extended in database literature for distinct value estimation,^{8, 14} but this work does not consider how human behaviors impact the sampling process of crowdsourced queries.

Species estimation techniques have been explored for search and meta-search engines. In Broder et al.² the authors develop an algorithm to estimate the size of any set of documents defined by certain conditions based on previous queries. Liu et al.¹⁷ describes an algorithm to estimate the corpus size for a meta-search engine to better direct queries to search engines. Similar techniques are also used to measure the quality of search engines.¹ Recent work explores species estimation techniques for the deep web,¹⁸ however, it does not consider the specific worker behavior and assumes sampling with replacement.

8. 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 crowd-sourced 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. By calculating an estimate of the expected result set size, or cardinality, for the enumeration query, an estimate of how complete the set is can be formed.

The species estimation literature provides a starting point to tackle cardinality estimation. However, applying existing estimators to sequences of responses from the crowd yields inaccurate results.

Instead of a with-replacement sample from a single item distribution, crowdsourced enumeration is a with-replacement sampling process drawing from many without-replacement sampling processes over potentially different distributions.

A particularly troublesome issue is the presence of “streakers,” workers who complete many more HITs than other workers, causing the estimator to wildly overestimate cardinality. To ameliorate the problems caused by these overzealous workers, we develop a *streaker-tolerant* estimator. Using enumeration experiments run on AMT, we show that this estimator successfully reduces the impact of streakers and generates a more accurate estimate.

In this paper, we investigated one common operation used in a RDBMS: enumerating the items from a crowdsourced relation. Of course, modern database systems support many other operations, called relational operators, that can be composed to form more expressive queries.

We believe the result set size estimation techniques we described can be used with these more expressive queries. Some operators will be simple to apply as a subsequent processing step on the result of the crowdsourced enumeration. If each item provided by the crowd is processed to produce a single item, it is straightforward to apply our cardinality estimation techniques to reason about completeness of the final query result. An example is the `PROJECT` operator, which returns to the user only part of the information about each item, such as returning only the scientific name for each plant in the plants query. In contrast, consider the `JOIN` operator, which combines data from multiple relations, a process which typically matches pairs of items from two relations with respect to some predicate. The estimation of result set size in this case may need to incorporate statistics about the likelihood of items matching. Another common operation in RDBMSs is aggregation, for example, determining the average height of plants that can tolerate low light. The average could be calculated using the current set of responses received from crowd workers; cardinality estimation techniques may then be helpful in estimating the confidence of this average, or how close the current calculation is to the true value. Additionally, recent work has explored how to use crowd input to reduce the uncertainty of query answers over dirty and incomplete datasets for such operators.²³

Incorporating the crowd into query processing systems presents the opportunity to extend the usefulness of these systems, but also raises fundamental questions about the meaning of query results. The cardinality estimation methods we developed and described in this paper enable query progress estimation, thereby allowing users to make sense of query results from hybrid human/machine database

systems. By adapting statistical techniques, we enable users to reason about query completeness in the open world.

Acknowledgments

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 NSF graduate fellowship, in part by NSF CISE Expeditions award CCF-1139158, and gifts from Amazon, 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

- Bar-Yossef, Z., Gurevich, M. Efficient search engine measurements. *ACM Trans. Web* 5, 4 (Oct. 2011), 18:1–18:48.
- Broder, A., Fontura, M., Josifovski, V., Kumar, R., Motwani, R., Naber, S., Panigrahy, R., Tomkins, A., Xu, Y. Estimating corpus size via queries. In *Proceedings of CIKM* (2006).
- Bunge, J., Fitzpatrick, M. Estimating the number of species: A review. *J. Am. Stat. Assoc.* 88, 421 (1993), 364–373.
- Bunge, J., et al. Comparison of three estimators of the number of species. *J. Appl. Stat.* 22, 1 (1995), 45–59.
- Burnham, K.P., Overton, W.S. Estimation of the size of a closed population when capture probabilities vary among animals. *Biometrika* 65, 3 (1978), 625–633.
- Chao, A. Species estimation and applications. In *Encyclopedia of Statistical Sciences*, 2nd edn. N. Balakrishnan, C.B. Read, and B. Vidakovic, eds. Wiley, New York, 2005, 7907–7916.
- Chao, A., Lee, S. Estimating the number of classes via sample coverage. *J. Am. Stat. Assoc.* 87, 417 (1992), 210–217.
- Charikar, M., et al. Towards estimation error guarantees for distinct values. In *Proceedings of the PODS* (2000).
- Colwell, R.K., Coddington, J.A. Estimating terrestrial biodiversity through extrapolation. *Philos. Trans. Biol. Sci.* 345, 1311 (1994), 101–118.
- Doan, A., et al. Crowdsourcing applications and platforms: A data management perspective. *PVLDB* 4, 12 (2011), 1508–1509.
- Franklin, M.J., et al. CrowdDB: Answering queries with crowdsourcing. In *Proceedings of the SIGMOD* (2011).
- Good, I.J. The population frequencies of species and the estimation of population parameters. *Biometrika* 40, 3/4 (1953), 237–264.
- Gray, J., et al. Quickly generating billion-record synthetic databases. In *Proceedings of the SIGMOD* (1994).
- Haas, P.J., et al. Sampling-based estimation of the number of distinct values of an attribute. In *Proceedings of the VLDB* (1995).
- Heer, J., et al. Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. In *Proceedings of the CHI* (2010).
- Ipeirotis, P.G., Provost, F., Wang, J. Quality management on Amazon mechanical turk. In *Proceedings of the HCOMP* (2010).
- Liu, K.-L., Yu, C., Meng, W. Discovering the representative of a search engine. In *Proceedings of the CIKM* (2002).
- Lu, J., Li, D. Estimating deep web data source size by capture–recapture method. *Inf. Retr.* 13, 1 (Feb. 2010), 70–95.
- Marcus, A., Wu, E., Madden, S., Miller, R. Crowdsourced databases: Query processing with people. In *Proceedings of the CIDR* (2011).
- Parameswaran, A., Polyzotis, N. Answering queries using humans, algorithms and databases. In *Proceedings of the CIDR* (2011).
- Shen, T., et al. Predicting the number of new species in further taxonomic sampling. *Ecology* 84, 3 (2003).
- Trushkowsky, B., Kraska, T., Franklin, M.J., Sarkar, P. Crowdsourced enumeration queries. In *Proceedings of the ICDE* (2013).
- Wang, J., et al. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the SIGMOD* (2014), 469–480.

Beth Trushkowsky (beth@cs.hmc.edu), Department of Computer Science, Harvey Mudd College, Claremont, CA.

Tim Kraska (tim.kraska@brown.edu), Department of Computer Science, Brown University, Providence, RI.

Michael J. Franklin (franklin@cs.berkeley.edu), Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA.

Purnamrita Sarkar (purna.sarkar@austin.utexas.edu), Department of Statistics and Data Sciences, University of Texas at Austin, Austin, TX.

© 2016 ACM 0001-0782/16/01 \$15.00

Coming Next Month in
COMMUNICATIONS

**YFCC100:
The New Data on
Multimedia Research**

**Phonetic Analytics
Technology and Big Data**

**Accountability in
Algorithmic Decision-
Making**

**The Verification
of a Distributed System**

**Schema.org: Evolution
of Structured Data
on the Web**

**Plus the latest news about defending the Internet,
self-healing robots, and Internet privacy.**

**The Beekman Report on
Database Research**

**Lessons from
the Tech Transfer Trenches**

**A Conversation about
Bulk Surveillance**

**Turing Lecture:
The Land Sharks
Are on the Squawk Box**

**Authentication Using
Pulse-Response Biometrics**

**Pinocchio: Nearly Practical
Verifiable Computation**