

Estimating Statistical Aggregates on Probabilistic Data Streams

T. S. Jayram
IBM Almaden Research
Almaden, CA
jayram@almaden.ibm.com

S. Muthukrishnan
Google Research
New York, NY
muthu@google.com

Andrew McGregor
ITA Center, UCSD
San Diego, CA
andrewm@ucsd.edu

Erik Vee*
Yahoo! Research
Sunnyvale, CA
erikvee@yahoo-inc.com

ABSTRACT

The probabilistic-stream model was introduced by Jayram et al. [20]. It is a generalization of the data stream model that is suited to handling “probabilistic” data, where each item of the stream represents a probability distribution over a set of possible events. Therefore, a probabilistic stream determines a distribution over a potentially exponential number of classical “deterministic” streams where each item is deterministically one of the domain values.

Designing efficient aggregation algorithms for probabilistic data is crucial for handling uncertainty in data-centric applications such as OLAP. Such algorithms are also useful in a variety of other settings including analyzing search engine traffic and aggregation in sensor networks.

We present algorithms for computing commonly used aggregates on a probabilistic stream. We present the first one pass streaming algorithms for estimating the expected mean of a probabilistic stream, improving upon results in [20]. Next, we consider the problem of estimating frequency moments for probabilistic data. We propose a general approach to obtain unbiased estimators working over probabilistic data by utilizing unbiased estimators designed for standard streams. Applying this approach, we extend a classical data stream algorithm to obtain a one-pass algorithm for estimating F_2 , the second frequency moment. We present the first known streaming algorithms for estimating F_0 , the number of distinct items on probabilistic streams. Our work also gives an efficient one-pass algorithm for estimating the median of a probabilistic stream.

Categories and Subject Descriptors

F.2 [Analysis of Algorithms & Problem Complexity]

*Part of this work was done when the author was at IBM Almaden Research Center.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODS'07, June 11–14, 2007, Beijing, China.

Copyright 2007 ACM 978-1-59593-685-1/07/0006 ...\$5.00.

General Terms

Algorithms, Design, Performance, Theory

Keywords

probabilistic streams, OLAP, mean, median, frequency moments

1. INTRODUCTION

In many applications it is becoming increasingly necessary to consider uncertain data, e.g., information that is incomplete, unreliable, or noisy. In this paper, we focus on efficiently calculating aggregation measures over uncertain data, with a particular emphasis on the OLAP model. OLAP is a multidimensional model of data in which the dimensions are structured hierarchically, and *facts* map to points in the corresponding multidimensional space. Dealing with uncertain information in OLAP is widely recognized to be an important problem and has received increasing attention recently. A particular kind of uncertainty, called *imprecision*, arises when facts do not map to points but to regions consistent with the domain hierarchies associated with dimension attributes. For example, we can denote that a particular repair took place in the state California without specifying a city.

Motivated by several criteria, [7] proposed the following solution. Using a mathematically principled approach, each imprecise “region” fact r in the database is replaced with a set of precise “point” facts representing the possible completions of r , together with their probabilities. For example, a fact about California might be replaced by 3 facts: one about Los Angeles, one about San Diego, and one about San Francisco, with associated probabilities 0.3, 0.2, 0.5, respectively. This defines a sample space of possible worlds; the probability associated with each world can be computed easily. The answer to an aggregation query Q is then defined to be the expected value of Q over the possible worlds. A similar data model was considered earlier by [16] in the context of non-aggregation queries, such as conjunctive queries, wherein an algebra of atomic events is used to define the semantics of queries. However, the data (expression) complexity of evaluating conjunctive queries in this framework has been shown to be $\#P$ -complete [12].

To answer queries *efficiently*, we cannot explicitly enumerate all the possible worlds, since in the worst case, this

could be exponentially large in the size of the database. A more significant challenge while dealing with OLAP queries arises when aggregation queries are qualified by specifying the dimension values at some level of granularity, e.g., average of all repair costs in San Francisco. Here, even the set of facts that match the query specification is a varying quantity over the possible worlds. Furthermore, because of the massive amounts of data being analyzed, it is impractical to process the data using many random accesses, necessitating algorithms that work over *probabilistic streams*.

1.1 Probabilistic Streams

In order to deal with massive data sets that arrive online and have to be monitored, managed and mined in real time, the *data stream* model has become popular. In stream A , new item a_t that arrives at time t is from some universe $[n] := \{1, \dots, n\}$. Applications need to monitor various aggregate queries such as quantiles, number of distinct items, heavy-hitters, etc. in small space, typically $O(\text{polylog } n)$. Data stream management systems are becoming mature at managing such streams and monitoring these aggregates on high speed streams such as in IP traffic analysis [11], financial and scientific data streams [3], and others. See [2, 23] for surveys of systems and algorithms for data stream management.

A generalization of the model, the *probabilistic stream model*, was recently introduced [20]. Here, each item a_t is not an element of the universe, but represents a distribution over elements of the universe together with a probability that the element is not actually present in the stream. The probabilistic stream model is applicable in a variety of settings in addition to the OLAP model. For example, a probabilistic stream may be derived from a stream of queries to a search-engine. Here a query “Stunning Jaguar” would be associated with different topics of interest, such as “Car”, “Animal”, or “Operating System”, with different probabilities. There are also instances when the input stream is probabilistic by its nature: for example, when we measure physical quantities such as light luminosity, temperature and others, we actually measure an average of these quantities over many tiny time instants in the analog world to get a single digital reading.

Previous work on probabilistic streams has included data stream algorithms for certain aggregate queries [6, 8, 20]. In this paper, we improve those results and study additional aggregate functions which are fundamental in any stream monitoring scenario. We note that some related results for estimating the expectation and variance of the number of distinct values and join/self-join sizes of a probabilistic stream have recently been proven independently of our work [10]. In what follows, we will first describe the model precisely and then state our results.

1.2 The Model and Definitions

In this section, we formally define the probabilistic stream model and the aggregate statistics that we will study.

DEFINITION 1 (PROBABILISTIC STREAM). *A probabilistic stream is a data stream $A = \langle \vartheta_1, \vartheta_2, \dots, \vartheta_m \rangle$ in which each data item ϑ_i encodes a random variable that takes a value in $[n] \cup \{\perp\}$. In particular each ϑ_i consists of a set of at most l tuples of the form (j, p_j^i) for some $j \in [n]$ and $p_j^i \in [0, 1]$. These tuples define the random variable X_i where*

$X_i = j$ with probability p_j^i and $X_i = \perp$ otherwise. We define $p_\perp^i = \Pr[X_i = \perp]$ and $p_j^i = 0$ if not otherwise determined.

A probabilistic stream naturally gives rise to a distribution over “deterministic” streams. Specifically we consider the i th element of the stream to be determined according to the random variable X_i and each element of the stream to be determined independently. Hence,

$$\Pr[\langle x_1, x_2, \dots, x_m \rangle] = \prod_{i \in [m]} \Pr[X_i = x_i] = \prod_{i \in [m]} p_{x_i}^i.$$

Throughout this paper we assume each probability specified in the probability stream is either 0 or $\Omega(1/\text{poly}(n))$.

We will be interested in the expected value of various quantities of the deterministic stream induced by a probabilistic stream. Computing the expected value is intuitively justified, since it is a natural way to summarize the answers of computing Q over an exponential number of possible streams. A formal justification using desiderata for handling imprecision in OLAP is given in [7].

DEFINITION 2 (AGGREGATION QUERIES). *We consider the following aggregate properties:*

1. **SUM** = $\mathbb{E} \left[\sum_{i \in [m]: X_i \neq \perp} X_i \right]$
2. **COUNT** = $\mathbb{E} [|\{i \in [m] : X_i \neq \perp\}|]$
3. **MEAN** = $\mathbb{E} \left[\frac{\sum_{i \in [m]: X_i \neq \perp} X_i}{|\{i \in [m] : X_i \neq \perp\}|} \mid |\{i \in [m] : X_i \neq \perp\}| \neq 0 \right]$
4. **DISTINCT** = $\mathbb{E} [|\{j \in [n] : \exists i \in [m], X_i = j\}|]$
5. **REPEAT-RATE** = $\mathbb{E} \left[\sum_{j \in [n]} |\{i \in [m] : X_i = j\}|^2 \right]$
6. **MEDIAN** = x such that $\lceil \text{COUNT} \rceil / 2$ is greater than $\max\{\mathbb{E} [|\{i \in [m] : X_i < x\}|], \mathbb{E} [|\{i \in [m] : X_i > x\}|]\}$

These are fundamental aggregates for stream monitoring, and well-studied in the classical stream model. For example, **REPEAT-RATE** in the classical model is the F_2 estimation problem studied in the seminal [1]. **DISTINCT** (also known as F_0) and **MEDIAN** have a long history in the classical data stream model [22, 15]. In the probabilistic stream model, we do not know of any prior algorithm with guaranteed bounds for these aggregates except the **MEAN** (**SUM** and **COUNT** are trivial) which, as we discuss later, we will improve.

As in classical data stream algorithms, our algorithms need to use $\text{polylog}(n)$ space and time per item, on a single pass over the data. It is realistic to assume that $l = O(\text{polylog } n)$ so that processing each item in the stream is still within this space and time bounds. In most motivating instances, l is likely to be quite small, say $O(1)$. In fact, it can be shown that $l = 1$ is sufficient for OLAP queries. Also, since the deterministic stream is an instance of the probabilistic stream (with each item in the probabilistic stream being one deterministic item with probability 1), known lower bounds for the deterministic stream model carry over to the probabilistic stream model. As a result, for most estimation problems, there does not exist streaming algorithms that precisely calculate the aggregate. Instead, we will focus on returning approximations to the quantities above. We say a value \hat{Q} is an (ε, δ) -approximation for a real number Q if $|\hat{Q} - Q| \leq \varepsilon Q$ with probability at least $(1 - \delta)$.

over its internal coin tosses. Many of our algorithms will be deterministic and return $(\epsilon, 0)$ -approximations. When approximating **MEDIAN** it makes more sense to consider a slightly different notion of approximation. We say x is an ϵ -approximate median if $(1/2 + \epsilon) \lceil \text{COUNT} \rceil$ is greater than

$$\max\{\mathbb{E}[\lceil \{i \in [m] : X_i < x\} \rceil], \mathbb{E}[\lceil \{i \in [m] : X_i > x\} \rceil]\}.$$

1.2.1 Related Models

The probabilistic stream model complements a stream model that has been recently considered where the stream consists of independent samples drawn from a single unknown distribution [9, 18]. In this alternative model, the probabilistic stream $A = \langle \vartheta_1, \dots, \vartheta_1 \rangle$ consists of a repeated element encoding a single probability distribution over $[n]$, but the crux is that the algorithm does not have access to the probabilistic stream. The challenge is to infer properties of the probability distribution a_1 from a randomly chosen deterministic stream. There is related work on reconstructing strings from random traces [5, 21]. Here, each element of probabilistic stream is of the form $\{(i, 1 - p), (\perp, p)\}$ for some $i \in [n]$. As before, the algorithm does not have access to the probabilistic stream but rather tries to infer the probabilistic stream from a limited number of independently generated deterministic streams. The results in [5, 9, 18, 21] do not provide any bounds for estimating aggregates such as the ones we study here when input is drawn from multiple, known probability distributions.

1.3 Our Results

We present the first single-pass algorithms for estimating the aggregate properties **MEAN**, **MEDIAN**, **REPEAT-RATE**, and **DISTINCT**. The algorithms for **MEAN** and **MEDIAN** are deterministic while the other two algorithms are randomized. While it is desirable for all the algorithms to be deterministic, this randomization can be shown to be necessary using the standard results in streaming algorithms, eg., of Alon, Matias, and Szegedy [1, Proposition 3.7]. Our results are summarized in the Table 1.

At first glance, **MEAN** seems trivial to estimate and this is the case with deterministic streams. **SUM** and **COUNT** can indeed be estimated easily since they are linear in the input and, hence, we can write straightforward formulas to compute them. However **MEAN** is not simply **SUM/COUNT** as shown in [20] and needs nontrivial techniques.

We present two single-pass, deterministic, $(\epsilon, 0)$ -approx algorithms for **MEAN**. The first using $O(\log \epsilon^{-1})$ space while the second, using an alternative technique, uses $O(\epsilon^{-1/2})$ space. Furthermore, if **COUNT** is sufficiently large then we present a simple algorithm that only needs $O(1)$ words of space. The best known previous work [20] presents an $O(\log m)$ pass algorithm using $O(\epsilon^{-1} \log^2 m)$ space. Thus our algorithms represent a considerable improvement.

Other aggregates such as **MEDIAN**, **REPEAT-RATE**, and **DISTINCT** have previously known algorithms for deterministic streams. A natural approach therefore would be to randomly instantiate multiple deterministic streams, apply standard stream algorithms and then apply a robust estimator. This approach works to an extent but typically gives poor approximations to small quantities. This is the case for **DISTINCT**. Furthermore, this approach necessitates a high degree of randomization. For **MEDIAN**, we show that it is possible to deterministically instantiate a single deterministic stream and run standard stream algorithms. We show

that this approach also works for **REPEAT-RATE** in the full version of this paper but there is a slightly more efficient method which we now describe.

We consider the problem of evaluating frequency moments on probabilistic data. (Recall that the i -th frequency moment F_i for an input stream of elements from a fixed domain of size n is defined to be the sum of the i -th power of the frequencies of the different elements in the stream. The extension to probabilistic data is defined naturally using possible worlds.) Data stream algorithms for frequency moments are well-studied and use sophisticated techniques based on hashing to produce high quality estimates. We first extend previous techniques to find a probabilistic stream algorithm for F_0 (i.e. **DISTINCT**). We next show that F_2 (i.e. **REPEAT-RATE**) on probabilistic streams can be computed in one pass using space and update time of $O(\epsilon^{-2}(\log n + \log m))$, matching the resources used by the well-known data stream algorithm for F_2 [1]. Our algorithm exploits some simple but powerful features of the classical data stream algorithm for F_2 . We make the general observation that for any aggregator, an unbiased estimator for that aggregator over streams is also an unbiased estimator for that aggregator over probabilistic streams. Now, the basic estimator for F_2 simply hashes the stream and then applies a simple aggregation algorithm on the hashed input. We design a simple algorithm for this very same aggregator on probabilistic streams. Although this technique works in general, we have no *a priori* guarantee that estimators found in this way will have small variance in the probabilistic-stream setting. Still, we show that the variance associated with the estimator for F_2 is small by analyzing its algebraic structure. After demonstrating that the variance is small, we appeal to a standard set of techniques to get the estimate to within the desired relative error with high confidence.

We finish by considering the problems of answering OLAP-style queries involving *roll-ups* and *drill-downs* in a more efficient manner. Traditionally this is achieved by keeping a small set of sufficient statistics for queries at lower levels of the dimension hierarchy so that they can be combined to answer queries at higher levels of the hierarchy. Ideally, the size of this set should be independent of the size of the database. For **MEAN**, we show that it suffices to keep a $O(\log \epsilon^{-1})$ -size set of sufficient statistics in order to answer such related queries. For F_2 , it also suffices to keep a small set of sufficient statistics; for this case, the set-size is $O(\epsilon^{-2})$. For **MIN/MAX**, we show that the algorithm [20] in our previous paper yields a set of $O(\epsilon^{-1})$ sufficient statistics.

2. ESTIMATING MEAN

As has been noted previously in [7, 20], this is more technically challenging than might be expected. The naïve approach, simply computing **SUM** and **COUNT** and looking at their quotient, fails to be a good estimator in even simple examples. Consider the example from [20], in which we have a probabilistic stream $\{(1, 1.0)\}, \{(10, 0.1)\}$. The true value of **MEAN** is 1.45, while the value of **SUM/COUNT** is approximately 1.81. Despite this, we will that **SUM/COUNT** is a good approximation when **COUNT** is sufficiently large.

The difficulty in computing **MEAN** arises in part since we must divide by the number of items actually occurring in the stream, a quantity that varies over different possible worlds. The approach taken in [20] was to rewrite the expression for the expected value of the average in terms of a generating

	Space	Randomized/Deterministic
MEAN	$O(\log \varepsilon^{-1})$	Deterministic
DISTINCT	$O(\varepsilon^{-5} \log n \log^2 \delta^{-1})$	Randomized
REPEAT-RATE	$O(\varepsilon^{-2} (\log n + \log m) \log \delta^{-1})$	Randomized
MEDIAN	$O(\varepsilon^{-2} \log m)$	Deterministic

Table 1: Streaming Algorithms for Probabilistic Streams

function. Taking a derivative of this expression eliminated the division, but then required us to estimate an integral.

Their computation was based on the observation that only two relevant values were needed for each probabilistic item (i.e., sufficient statistics): p_i , the probability that item i is not \perp ; and a_i , the expected value of item i , given that it is not \perp . They then show the following.

PROPOSITION 1 ([20]). **MEAN** can be rewritten as

$$\text{MEAN} = \int_0^1 \psi(x) dx$$

where $\psi(x) = \sum_{i \in [m]} a_i p_i \cdot \prod_{j \neq i} (1 - p_j + p_j x)$.

We note that their definition of **MEAN** assumed that at least one item in the stream appeared with probability 1. Under our definition, we divide the value of **MEAN** given above by the probability that the stream contains at least one element, i.e. $1 - \prod_{i \in [m]} p_i$. For readability, we will assume throughout Sections 2.1 and 2.2 that $p_i = 1$ for some i (hence, **COUNT** ≥ 1).

Their approach led to an $O(\log m)$ -pass algorithm that estimated the integral; each pass required $O(\varepsilon^{-1} \log m)$ update time per item. Although this may be adequate for off-line applications, it is simply too slow for on-line applications with large m and small ε . Here, we provide a one-pass algorithm with update time $O(\log \varepsilon^{-1})$, an exponential improvement.

Our approach uses a careful, technically-involved analysis of the integral. We might hope that a simple Taylor-series expansion for ψ would give us a good estimate. Unfortunately, this approach fails. (To illustrate this, take $a_i = 1, p_i = 1/2$ for all i . Note that $\psi(x)$ is only interesting near $x = 1$, while the coefficients of the Taylor series expansion about $x = 1$ grow exponentially.) However, by properly rewriting the integral, we get a step closer. Let $I \subseteq [m]$, and define

$$g_I(z) = \prod_{i \in I} (1 - p_i z), \quad h_I(z) = \sum_{i \in I} \frac{a_i p_i}{1 - p_i z}, \quad f_I(z) = g_I(z) h_I(z)$$

Notice that $f_I(z)$ is actually a polynomial, and further, $f_I(1 - x) = \psi(x)$ when $I = [m]$. Applying a change of variable to the integral in Proposition 1 (setting $z = 1 - x$), and rewriting somewhat, we see that

$$\begin{aligned} \text{MEAN} &= \int_0^1 g_I(z) g_J(z) (h_I(z) + h_J(z)) dz \\ &= \int_0^1 g_I(z) (h_I(z) g_J(z) + f_J(z)) dz, \end{aligned}$$

where $J = [m] - I$. Not surprisingly, it will be easier to approximate each of these functions when z is bounded away from 1. Let $P = \sum_i p_i = \text{COUNT}$, and for any $\varepsilon < 1/2$,

define $z_0 = \min\{1, \frac{1}{P} \ln(2P/\varepsilon)\}$. (Recall $P \geq 1$, so $z_0 > 0$.) We have the following lemma. Its proof can be found in the full version of this paper.

LEMMA 2. Let $\varepsilon < 1/20$, with $P \geq 1$ and z_0 defined as above. Then for any $I \subseteq [m]$, $J = [m] - I$,

$$(1 - \varepsilon) \text{MEAN} \leq \int_0^{z_0} g_I(z) (h_I(z) g_J(z) + f_J(z)) dz \leq \text{MEAN}.$$

One of our key insights is that, whereas approximating the integrand directly is difficult, approximating each of the functions individually can be done with low-degree polynomials. A second key insight is that, while approximating $h_I(z)$ can be done directly with a Taylor-series expansion, we need to approximate the *logarithm* of $g_I(z)$. If z_0 is sufficiently small, say $z_0 \leq 1/e$, then we set $I = [m]$, $J = \emptyset$; the approximations of $\ln g_I(z)$ and $h_I(z)$ can be written as Taylor series with quickly converging remainder terms. However, if z_0 is near 1 (which happens when P is small), then we need an extra trick. In this case, we set

$$I = \{i \in [m] : p_i < 1/e\} \text{ and } J = [m] - I.$$

In this case, the approximations of $\ln g_I(z)$ and $h_I(z)$ will converge because the coefficients will be tiny. Further, we will show that $|J|$ is necessarily small, since P must be small. So $f_J(z), g_J(z)$ are already low-degree polynomials; we simply calculate them. Amazingly, the degree of the polynomials is independent of m .

In fact, when P is very large (and $I = [m]$), $h_I(z)$ is approximately **SUM**, while $\ln g_I(z)$ is approximately $-Pz$ (i.e. polynomials of degree 0 and 1, respectively). Applying this argument more carefully allows us to bound the error of using **SUM**/ $P = \text{SUM}/\text{COUNT}$ to approximate **MEAN**.

With this approach, we can find good approximations to $g_I(z)$ and $h_I(z)$. But we are not quite done. Our approximation to $h_I(z)$ is a low-degree polynomial, while our approximation to $g_I(z)$ is exp(low-degree polynomial). There is no closed-form evaluation for the integral of the product of these two expressions. So we approximate our *approximation* of $g(z)$, as a polynomial. The details appear in the next subsections.

2.1 Long Streams

In this subsection, we show that **SUM**/**COUNT** is a good approximation to **MEAN** when $P = \text{COUNT}$ is large. The techniques we use here will serve as a warm-up to the more involved analysis of the next subsection. We have the following.

THEOREM 1. For $P \geq e$,

$$\frac{\text{SUM}}{\text{COUNT}} \left(1 - \frac{2 \ln P}{P}\right) \leq \text{MEAN} \leq \frac{\text{SUM}}{\text{COUNT}} \left(1 + \frac{1}{P-1}\right)$$

PROOF. We first prove the easier inequality.

$$\begin{aligned} \text{MEAN} &= \sum_{i \in [m]} a_i p_i \int_0^1 \prod_{j \neq i} (1 - p_j z) dz \\ &\leq \sum_{i \in [m]} a_i p_i \int_0^1 e^{-z \sum_{j \neq i} p_j} dz. \end{aligned}$$

Now, for any i , we have

$$\int_0^1 e^{-z \sum_{j \neq i} p_j} dz \leq \int_0^1 e^{-z(P-1)} dz = \left(\frac{e^{-z(P-1)}}{-(P-1)} \right) \Big|_0^1 \leq \frac{1}{P-1}$$

Recalling that $P = \text{COUNT}$, we have

$$\text{MEAN} \leq \sum_{i \in [m]} a_i p_i \frac{1}{P-1} = \frac{\text{SUM}}{\text{COUNT}} \left(1 + \frac{1}{P-1} \right)$$

For the other inequality, let $\delta \in [0, 1]$. We use the fact that $h(z) = \sum_{i \in I} a_i p_i (1 - p_i z)^{-1} \geq \text{SUM}$ for all $z \geq 0$. (Again, $I = [m]$ for now.) Further, noting that $1 - y \geq (1 - \delta)^{y/\delta}$ for $y \in [0, \delta]$, we get that for $z \leq \delta$,

$$g(z) = \prod_{i \in I} (1 - p_i z) \geq \prod_{i \in I} (1 - \delta)^{p_i z/\delta} = (1 - \delta)^{Pz/\delta}.$$

Putting this together, we have

$$\begin{aligned} \text{MEAN} &= \int_0^1 g(z) h(z) dz \geq \int_0^\delta (1 - \delta)^{Pz/\delta} \cdot \text{SUM} dz \\ &= \frac{\text{SUM}}{\text{COUNT}} \cdot \frac{\delta}{\ln(\frac{1}{1-\delta})} (1 - (1 - \delta)^P) \end{aligned}$$

Examining the Taylor series, we can show that $-\ln(1 - \delta) \leq \delta + \delta^2$ for $\delta < 1/2$. Hence,

$$\frac{\delta}{\ln(\frac{1}{1-\delta})} \geq \frac{1}{1 + \delta} \geq 1 - \delta \quad \text{for } \delta < 1.$$

So we have, for $\delta \leq 1/2$,

$$\begin{aligned} \text{MEAN} &\geq \frac{\text{SUM}}{\text{COUNT}} (1 - \delta) (1 - (1 - \delta)^P) \\ &\geq \frac{\text{SUM}}{\text{COUNT}} (1 - \delta) (1 - e^{-\delta P}). \end{aligned}$$

Setting $\delta = \ln P/P$, and noting $\ln P/P \leq 1/e$ for $P \geq e$, we have for $P \geq e$,

$$\text{MEAN} \geq \frac{\text{SUM}}{\text{COUNT}} (1 - 2 \ln P/P).$$

□

With some algebra, we have the following corollary.

COROLLARY 3. For any $\varepsilon < 1/20$, if $P \geq \frac{4}{\varepsilon} \ln(2/\varepsilon)$, then

$$(1 - \varepsilon) \frac{\text{SUM}}{\text{COUNT}} \leq \text{MEAN} \leq (1 + \varepsilon) \frac{\text{SUM}}{\text{COUNT}}$$

Therefore, in the case that we know COUNT (i.e. P) is very large, or when we are not concerned with estimating MEAN extremely well, then the simple method of using SUM/COUNT works well. However, when COUNT is small or we need to be very accurate then using SUM/COUNT provides an inadequate estimate. We address this in the next subsection.

2.2 Short Streams

We now give an extremely efficient one-pass streaming algorithm that estimates MEAN within a multiplicative $(1 + \varepsilon)$ factor. Remarkably, both the update time, $O(\log(1/\varepsilon))$, and the number of memory registers, $O(\log(1/\varepsilon))$, are independent of m , the number of items in the stream. It is also interesting that the algorithm is entirely deterministic, so it is guaranteed to return a good estimate. In contrast to typical streaming algorithms, there is no chance of failure.

Assume that $\varepsilon < 1/20$. We maintain

$$\sum_{i \in [m]} p_i^k \quad \text{and} \quad \sum_{i \in [m]} a_i p_i^k \quad \text{for } k \in [O(\log(1/\varepsilon))],$$

where the exact values for k are given below. In addition, we also remember the values a_j, p_j for every j such that $p_j > 1/e$, until $P \geq 6 \ln(2/\varepsilon)$. (Notice that we will need to remember at most $6e \ln(2/\varepsilon)$ such values.) Using these maintained values, we can quickly compute a $(1 + 5\varepsilon)$ -approximations for MEAN .

Somewhat more formally, let $I = \{i \in [m] : p_i \leq 1/e\}$ if $P < 6 \ln(2/\varepsilon)$, and let $I = [m]$ otherwise. Let

$$P_k = \sum_{i \in I} p_i^k, \quad A_k = \sum_{i \in I} a_i p_i^k,$$

$$f_J(z) = \sum_{i \in J} a_i p_i \prod_{\substack{j \in J \\ j \neq i}} (1 - p_j z), \quad \text{and} \quad g_J(z) = \prod_{j \in J} (1 - p_j z)$$

where $J = [m] - I$. Note that given the maintained values, it is easy to calculate $f_J(z)$, $g_J(z)$ and P_k, A_k for $k = 1, 2, \dots, O(\log(1/\varepsilon))$. We have our main result:

THEOREM 2. Let $\varepsilon < 1/20$, and define P_k, A_k, I, J, f_J , and g_J as above. Let ℓ be an even integer greater than or equal to $8 \ln(2P/\varepsilon)$, and define

$$\widetilde{\text{MEAN}} = \frac{1}{1 - \varepsilon} \cdot \frac{\text{SUM}}{\text{COUNT}}$$

if $P \geq \frac{4}{\varepsilon} \ln(2/\varepsilon)$, and otherwise define $\widetilde{\text{MEAN}}$ as

$$\frac{1}{1 - \varepsilon} \int_0^{z_0} \prod_{i=1}^{k_0} \left(\sum_{j=0}^{\ell} \frac{(-P_i z^i)^j}{j! \cdot i^j} \right) \left(f_J(z) + g_J(z) \sum_{i=1}^{k_1} A_{i+1} z^i \right) dz$$

where $k_0 = 3 \ln(2/\varepsilon)$, $k_1 = \ln(2/\varepsilon)$. Then

$$\text{MEAN} \leq \widetilde{\text{MEAN}} \leq (1 + 5\varepsilon) \text{MEAN}.$$

The corresponding probabilistic stream algorithm has update time $O(\log(1/\varepsilon))$ and uses $O(\log(1/\varepsilon))$ memory registers (assuming each register can hold an arbitrary real number).

We now proceed with the proof. Define

$$\tilde{g}_k(z) = \exp \left(- \sum_{j=1}^k \frac{P_j}{j} z^j \right), \quad \tilde{h}_k(z) = \sum_{j=0}^k A_{j+1} z^j$$

The following lemma verifies that these are good approximations to g_I, h_I , respectively. The key idea is that for large P , z_0 is small, while for small P , P_i and A_i decrease as e^{-i} . Its proof appears in the full version of the paper.

LEMMA 4. Define \tilde{g}_{k_0} and \tilde{h}_{k_1} as above, and let $k_0 \geq \ln(2P/\varepsilon)$ and $k_1 \geq \ln(2/\varepsilon)$. Then

$$g(z) \leq \tilde{g}_{k_0}(z) \leq (1 + \varepsilon)g(z), \quad h(z) \leq \tilde{h}_{k_1}(z) \leq (1 + \varepsilon)h(z)$$

Thus we have reduced the problem to that of estimating

$$\int_0^{z_0} \tilde{g}_{k_0}(z)(f_J(z) + g_J(z)\tilde{h}_{k_1}(z))dz$$

where $k_0 = \ln(2P/\varepsilon)$ and $k_1 = \ln(2/\varepsilon)$. Unfortunately, \tilde{g}_{k_0} is not a polynomial. So we now expand it.

LEMMA 5. *Let ℓ be the smallest even integer greater than $8 \ln(2P/\varepsilon)$. Then for $z \leq z_0$,*

$$\tilde{g}_{k_0}(z) \leq \prod_{i=1}^{k_0} \left(\sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} \right) \leq (1 + \varepsilon) \tilde{g}_{k_0}(z)$$

PROOF. We first expand $\exp(-P_i z^i/i)$.

$$\exp(-P_i z^i/i) = \sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} + R_i(z),$$

where $R_i(z) = \sum_{j \geq \ell+1} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j}$. We see that regardless of the value of z_0 , we have $P_i z^i \leq \ln(2P/\varepsilon)e^{-i+1}$: When $z_0 \leq 1/e$, then $P_i z^i \leq (P z_0) z_0^{i-1} \leq \ln(2P/\varepsilon)e^{-i+1}$, and if $z_0 \geq 1/e$, then $P_i z^i \leq P e^{-i+1} z^i \leq \ln(2P/\varepsilon)e^{-i+1}$. Since $\ell \geq \ln(2P/\varepsilon)$, the absolute values of the terms of $R_i(z)$ are decreasing. Since the series is alternating and ℓ is even, we see for $z \leq z_0$,

$$\begin{aligned} 0 \leq -R_i(z) &\leq \frac{1}{(\ell+1)!} \left(\frac{e^{-i+1} \ln(2P/\varepsilon)}{i} \right)^{\ell+1} \\ &\leq \left(\frac{e^{-i+1} e \ln(2P/\varepsilon)}{i(\ell+1)} \right)^{\ell+1} \end{aligned}$$

So we have for $z \leq z_0$,

$$\begin{aligned} -R_i(z) &\leq \left(\frac{e^{-i+1} e \ln(2P/\varepsilon)}{i} \right)^{\ell+1} \leq \left(\frac{e^{-i} e^2}{i} \right)^{8 \ln(2P/\varepsilon)} \\ &\leq \left(\frac{\varepsilon}{2P} \right)^{8i} \left(\frac{1}{i} \right)^{8 \ln(2P/\varepsilon)} \end{aligned}$$

Noting that $\varepsilon/(2P) \leq e^{-P z_0}$ and bounding $(\varepsilon/(2P))^{8i-1} i^{-8}$ by $\varepsilon/(i 2^i)$, we have

$$\begin{aligned} 0 \leq -R_i(z) &\leq \left(\frac{\varepsilon}{2P} \right)^{8i} \left(\frac{1}{i} \right)^8 \leq \frac{1}{i^8} \left(\frac{\varepsilon}{2P} \right)^{8i-1} e^{-P z_0} \\ &\leq \frac{\varepsilon}{i 2^i} e^{-P_i z^i/i} \end{aligned}$$

First, using $R_i(z) \leq 0$, we have

$$\begin{aligned} \tilde{g}_{k_0}(z) &= \exp\left(-\sum_{i=1}^{k_0} P_i z^i/i\right) = \prod_{i=1}^{k_0} \exp(-P_i z^i/i) \\ &\leq \prod_{i=1}^{k_0} \left(\sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} \right). \end{aligned}$$

For the other side, we use the following claim, whose proof appears in the full version of this paper.

CLAIM 6. *For all $k > 0$ and for $\varepsilon < 1/2$,*

$$\prod_{i=1}^k \left(1 + \frac{1}{i 2^i} \varepsilon\right) \leq \left(1 + \frac{k}{k+1} \varepsilon\right).$$

Combining our claim and the fact that $-R_i(z) \leq \frac{\varepsilon}{i 2^i} e^{-P_i z^i/i}$, we have

$$\begin{aligned} \prod_{i=1}^{k_0} \left(\sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} \right) &\leq \prod_{i=1}^{k_0} \left(\left(1 + \frac{1}{i 2^i} \varepsilon\right) e^{-P_i z^i/i} \right) \\ &\leq (1 + \varepsilon) \prod_{i=1}^{k_0} e^{-P_i z^i/i} \\ &= (1 + \varepsilon) \tilde{g}_{k_0}(z) \end{aligned}$$

□

To complete the proof of the theorem, simply note that when $P \geq (4/\varepsilon) \ln(2/\varepsilon)$, the algorithm uses SUM/COUNT to approximate MEAN (which, by Corollary 3, is a good approximation). Hence, we may assume $k_0 = \ln(2P/\varepsilon) \leq \ln(2/\varepsilon) + \ln(4/\varepsilon) + \ln \ln(2/\varepsilon)$, which is less than $3 \ln(2/\varepsilon)$ when $\varepsilon < 1/20$. The error bound of $(1 + 5\varepsilon)$ comes from combining the errors from each of our approximations.

2.3 Short Streams (Alternative Approach)

In this section we briefly discuss an alternative idea for short stream that may be of separate interest, although the space and time bounds are not as good as those of the algorithm in the previous section. For the duration of this section let Y and Z be the random variables defined by,

$$Y = \sum_{i: X_i \neq \perp} X_i \quad \text{and} \quad Z = |\{i \in [m] : X_i \neq \perp\}|.$$

The main idea behind our algorithm is that if COUNT is not large then it is sufficient to estimate MEAN from $\Pr[Z = z]$ and $\mathbb{E}[Y|Z = z]$ for a relatively small range of values of z .

The next lemma shows that it is possible compute the probability that Z takes various values if we assume that at each point during the evolution of the stream, the number of elements that have appeared does not differ significantly from its expected number.

LEMMA 7. *For $j \in [m]$, let $Z_j = |\{i \in [j] : X_i \neq \perp\}|$ and $Y_j = \sum_{i \leq j: X_i \neq \perp} X_i$. Let C_j^w be the event that,*

$$\forall i \in [j], \quad |Z_i - \mathbb{E}[Z_i]| \leq w.$$

It is possible to compute $A_z = \Pr[Z = z, C_m^w]$ and $B_z = \sum_y y \Pr[Y = y, Z = z, C_m^w]$ for all z in a single pass using $O(w \log n)$ bits of space.

PROOF. We start by defining $A_{j,z} = \Pr[Z_j = z, C_j^w]$ and $B_{j,z} = \sum_y y \Pr[Y_j = y, Z_j = z, C_j^w]$. First note that for $j, z \in [m]$,

$$\begin{aligned} A_{j,z} &= \Pr[Z_{j-1} = z, X_j = \perp, C_j^w] \\ &\quad + \Pr[Z_{j-1} = z-1, X_j \neq \perp, C_j^w] \\ &= \begin{cases} A_{j-1,z-1}(1 - p_{\perp}^j) + A_{j-1,z} p_{\perp}^j & \text{if } |z - \mathbb{E}[Z_j]| \leq w \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where $A_{0,z} = 1$ if $z = 0$ and 0 otherwise. Also, on the

assumption that $|z - \mathbb{E}[Z_j]| \leq w$,

$$\begin{aligned} B_{j,z} &= \sum_y y(\Pr[Y_{j-1} = y, Z_{j-1} = z, X_j = \perp, C_m^w]) \\ &\quad + \sum_{a \in [m]} \Pr[Y_{j-1} = y - a, Z_{j-1} = z - 1, X_j = a, C_m^w] \\ &= p_\perp^j B_{j-1,z} \\ &\quad + (1 - p_\perp^j)(\mathbb{E}[X_j | X_j \neq \perp] A_{j-1,z-1} + B_{j-1,z-1}) \end{aligned}$$

and $B_{0,z} = 0$ for all z . If $|z - \mathbb{E}[Z_j]| > w$ then clearly, $B_{j,z} = 0$. Lastly $\mathbb{E}[Z_j] = \mathbb{E}[Z_{j-1}] + (1 - p_\perp^j)$. Hence we can compute $(A_{j,z})_{0 \leq z \leq m}$, $(B_{j,z})_{0 \leq z \leq m}$ and $\mathbb{E}[Z_j]$ from $(A_{j-1,z})_{0 \leq z \leq m}$, $(B_{j-1,z})_{0 \leq z \leq m}$ and $\mathbb{E}[Z_{j-1}]$. The space bound follows because at most $O(w)$ of $(A_{j,z})_{0 \leq z \leq m}$ and $(B_{j,z})_{0 \leq z \leq m}$ are non-zero for any j . \square

Hence, if we choose w to be large enough such that C_m^w is a sufficiently high probability event then we can use the above lemma to obtain a deterministic algorithm for estimating MEAN. We show:

THEOREM 3. *There is a single-pass $(\varepsilon, 0)$ -approximation of MEAN that uses $O(\varepsilon^{-1/2}(\log n + \log m + \log \varepsilon^{-1}))$ space.*

PROOF. Let $c = \frac{4}{\varepsilon} \ln(8mn/\varepsilon)$. First, if $\text{COUNT} \geq c$ then by Corollary 3, SUM/COUNT is an $(\varepsilon, 0)$ approximation of MEAN. Alternatively assume that $\text{COUNT} \leq c$. Let $w = \sqrt{\varepsilon}c$. By an application of the Chernoff-Hoeffding and union bounds,

$$\begin{aligned} \Pr[C_m^w] &\geq 1 - \sum_{i \in [m]} \Pr[|Z_i - \mathbb{E}[Z_i]| > w] \\ &\geq 1 - 2m \exp(-\varepsilon c/3) \\ &\geq 1 - \varepsilon/4n \end{aligned}$$

$\sum_{z \geq 1} \frac{1}{z} \sum_y y \Pr[Y = y, Z = z, C_m^w]$ can be computed using $O(w \log n)$ space as described in Lemma 7. Also note that $\Pr[Z \neq 0]$ can easily be computed. Then,

$$\begin{aligned} &\left| \sum_{z \geq 1} \frac{1}{z} \sum_y y \frac{\Pr[Y = y, Z = z, C_m^w]}{\Pr[Z \neq 0]} - \text{MEAN} \right| \\ &= \sum_{y,z \geq 1} \frac{y}{z} \frac{|\Pr[Y = y, Z = z, C_m^w] - \Pr[Y = y, Z = z]|}{\Pr[Z \neq 0]} \\ &\leq n \frac{\Pr[\neg C_m^w \cap Z \neq 0]}{\Pr[Z \neq 0]} = n \Pr[\neg C_m^w | Z \neq 0] \leq \varepsilon. \end{aligned}$$

This yields an $(\varepsilon, 0)$ approximation because $\text{MEAN} \geq 1$. \square

2.3.1 Precision Issues

One remaining issue that needs to be addressed is the precision to which the various quantities are calculated. In Lemma 7, it was assumed that all the arithmetic could be performed exactly. However, it is very likely that, for some z, j both $A_{j,z}$ and $B_{j,z}$ can be exponentially small and yet non-zero. We can easily show that errors introduced by a lack of precision do not accumulate sufficiently to cause a problem. In particular, let us assume that the calculation of each $A_{j,z}$ is performed with the addition of error β . Then, given that $A_{j,z} = A_{j-1,z-1}(1 - p_\perp^j) + A_{j-1,z}p_\perp^j$, if it is non-zero, a simple induction argument shows that $A_{j,z}$ is computed with error at most

$$(p_\perp^j + 1 - p_\perp^j)(j-1)\beta + \beta = j\beta.$$

Similarly, $B_{j,z}$ is computed with error at most,

$$p_\perp^j n(j-1)^2 \beta + (1 - p_\perp^j)(n(j-1)\beta + n(j-1)^2 \beta) + \beta \leq j^2 n \beta.$$

Therefore $\beta = \varepsilon m^{-2} n^{-2}$ suffices to compute

$$\sum_{y,z} \frac{y}{z} \Pr[Y = y, Z = z, C_m^w]$$

with additive-error at most ε which translates into a relative-error since the quantity being estimated is $\Omega(1)$.

3. ESTIMATING MEDIAN

We present an algorithm for finding an ε -approximate MEDIAN. Perhaps surprisingly, our approach is simpler than that required for MEAN. It is based on a deterministic reduction of the problem to median finding in a deterministic stream. To solve this problem we use the algorithm of Greenwald and Khanna [17].

THEOREM 4. *There exists a single pass algorithm finding an ε -approximate MEDIAN in $O(\varepsilon^{-1} \log m)$ space.*

PROOF. The idea is use the selection algorithm of Greenwald, Khanna [17], on an induced stream as follows.

1. For each tuple (j, p_j^i) in ϑ_i , put $\lfloor 2mp_j^i \varepsilon^{-1} \rfloor$ copies of j in an induced stream A' .
2. Using the algorithm of Greenwald, Khanna [17], find an element l such that

$$\begin{aligned} &\max\{|\{i \in A' : 1 \leq i < l\}|, |\{i \in A' : l < i \leq n\}|\} \\ &\leq (1/2 + \varepsilon/2)|A'|. \end{aligned} \tag{1}$$

where $|A'|$ denotes the length of the stream A' .

Note that $p_j^i \geq \lfloor 2mp_j^i \varepsilon^{-1} \rfloor / (2m\varepsilon^{-1}) \geq p_j^i - \varepsilon/(2m)$. Therefore, dividing Eq. 1 by $2m\varepsilon^{-1}$ yields

$$\begin{aligned} &\max \left\{ \left(\sum_{1 \leq j < l, i \in [m]} p_j^i \right), \left(\sum_{l < j \leq n, i \in [m]} p_j^i \right) \right\} - \varepsilon/2 \\ &\leq (1/2 + \varepsilon/2) \lceil \text{COUNT} \rceil. \end{aligned}$$

The result follows since $\text{COUNT} > 0$. \square

4. ESTIMATING DISTINCT ITEMS

In this section we present a (ε, δ) -approximation algorithm for DISTINCT (sometimes known as F_0). In contrast to the algorithm for MEAN, a major part of our algorithm is to actually randomly instantiate numerous deterministic streams and to compute the average value of the number of distinct values in these streams. However, this general approach will only give an (ε, δ) approximation in small space if the expected number of distinct values is not very small. In the following theorem we show that it is possible to also deal with this case. Furthermore, the random instantiations will be performed in a slightly non-obvious fashion for the purpose of achieving a tight analysis of the appropriate probability bounds.

THEOREM 5. *We can (ε, δ) -approximate DISTINCT in a single pass and $O(\varepsilon^{-5} \log n \log^2 \delta^{-1})$ space.*

PROOF. First note that,

$$\text{DISTINCT} = \sum_{j \in [n]} \left(1 - \prod_{i \in [m]} (1 - p_j^i) \right).$$

Consider $\text{COUNT} = \sum_i (1 - p_{\perp}^i)$. Then,

$$e^{-\text{COUNT}} \text{COUNT} \leq 1 - \prod_{i \in [m], j \in [n]} (1 - p_j^i) \leq \text{DISTINCT},$$

and $\text{DISTINCT} \leq \text{COUNT}$. Hence if $\text{COUNT} \leq \ln(1 + \varepsilon)$ then COUNT is an $(\varepsilon, 0)$ approximation for DISTINCT . We now assume that $\text{COUNT} > \ln(1 + \varepsilon)$ and hence

$$\text{DISTINCT} \geq \ln(1 + \varepsilon) e^{-\ln(1 + \varepsilon)} \geq \varepsilon/2$$

assuming ε is sufficiently small.

Consider the following basic estimator X ,

1. For each tuple (j, p_j^i) in ϑ_i , put j in an induced stream A' with probability p_j^i
2. Compute an $(\varepsilon/3, \delta/(2c_1))$ approximation X of $F_0(A')$ using [4].

Note that the stream A' is not generated according to the distribution defined by the probabilistic stream because more than one item can be generated for a given ϑ_i . However, the expected number of distinct elements in A' is the same as the expected number of distinct elements in a stream generated by the probabilistic stream A . The reason for generating A' in this way is that we may argue that the events that i and j appear in A' are independent. This will be important in our analysis.

We will compute $c_1 = 3^3 \cdot 2\varepsilon^{-3} \ln(4/\delta)$ of these basic estimators and average them to produce our final estimator Y . The accuracy and probability of success of our algorithm follows from the following claim.

CLAIM 8. $\Pr[|Y - \text{DISTINCT}| \leq \varepsilon \text{DISTINCT}] \leq \delta$

PROOF. First we effectively assume that $F_0(A')$ can be computed exactly. Let Z be the random variable corresponding to the average of c_1 copies of $F_0(A')$. $\mathbb{E}[Z] = \text{DISTINCT}$. Note that $c_1 Z$ can be thought of as the sum of nc_1 independent boolean trials: c_1 trials corresponding to each $j \in [n]$ (some trials may have zero probability of success) and hence,

$$\begin{aligned} \Pr[|Z - \text{DISTINCT}| \leq (\varepsilon/3)\text{DISTINCT}] \\ &= \Pr[|c_1 Z - c_1 \text{DISTINCT}| \leq (\varepsilon/3)c_1 \text{DISTINCT}] \\ &\leq 2 \exp(-\varepsilon^2 c_1 \text{DISTINCT}/27) \\ &\leq 2 \exp(-\varepsilon^3 c_1/(2 \cdot 27)) \\ &\leq \delta/2. \end{aligned}$$

With probability at least $1 - c_1 \delta/(2c_1) = 1 - \delta/2$ each of the c_1 calls to the distinct element counter returns an $\varepsilon/3$ approximation. Hence with probability at least $1 - \delta/2$, $Y = (1 \pm \varepsilon/3)Z$. Therefore with probability at least $1 - \delta$, $Y \subset (1 \pm \varepsilon/3)Z \subset (1 \pm \varepsilon/3)^2 \text{DISTINCT} \subset (1 \pm \varepsilon) \text{DISTINCT}$ assuming $\varepsilon < 1/4$. \square

The space bound follows since each $O(\varepsilon^{-3} \log \delta^{-1})$ basic estimator requires $O((\varepsilon^{-2} + \log n) \log \delta^{-1})$ space [4].

5. ESTIMATING REPEAT-RATE

In this section, we address the problem of computing frequency moments on probabilistic data, with a particular application to F_2 , sometimes known as REPEAT-RATE. (Note that we already discussed F_0 in another context in Section 4.) For an input $\vec{a} = (a_1, a_2, \dots, a_m)$ over a domain $[n]$, let f_a denote the frequency of element a in \vec{a} . We define $F_k(\vec{a}) = \sum_a f_a^k$. (The aggregate $F_0(\vec{a})$ is defined to be the number of distinct elements in \vec{a} .)

We first observe that any unbiased estimator working over streams is also an unbiased estimator working over probabilistic streams. We assume that each item in the data stream, ϑ_i , is of the form $\{(a_i, p_i), (\perp, 1 - p_i)\}$. Let (\vec{a}, \vec{p}) denote a probabilistic stream where $\vec{a} = (a_1, a_2, \dots, a_m)$ and $\vec{p} = (p_1, p_2, \dots, p_m)$. Let h be a function (chosen randomly according to some distribution), and let $h(\vec{a})$ denote the vector obtained by applying h to each element in \vec{a} .

LEMMA 9. *Let Q and Q' be two aggregate functions such that $Q(\vec{b}) = \mathbb{E}_h[Q'(h(\vec{b}))]$ for any input \vec{b} over the base domain. Then, for any probabilistic stream*

$$Q(\vec{a}, \vec{p}) = \mathbb{E}_h[Q'(h(\vec{a}), \vec{p})].$$

That is, $Q'(h(\vec{a}), \vec{p})$ is an unbiased estimator of $Q(\vec{a}, \vec{p})$.

PROOF. For any possible world, let $\vec{b} = (b_1, \dots, b_m)$ denote the subset of \vec{a} that appears in that world. The premise of the lemma implies that $Q(\vec{b}) = \mathbb{E}_h[Q(h(\vec{b}))]$. Taking the expected value over possible worlds, and interchanging the expectation on the right hand side yields the lemma. \square

Thus, given an unbiased estimator for streams, our task reduces to finding an algorithm that efficiently computes the estimator over probabilistic streams. In fact, this technique is useful in deriving one-pass probabilistic stream algorithms for a variety of frequency moments. Although these algorithms have approximately the correct expectations, showing that they are tightly concentrated about their expected values must be done on an algorithm-by-algorithm basis. Here, we apply the method to find a one-pass probabilistic stream algorithm approximating F_2 with high-probability.

Let us first briefly review the classical algorithm for F_2 over streams [1]. Let \mathcal{H} denote a uniform family of 4-wise independent hash functions such that $h : [n] \rightarrow \{-1, +1\}$ for each $h \in \mathcal{H}$. In other words, $h(z_1), h(z_2), h(z_3)$, and $h(z_4)$ are independent over the random choice of h for any distinct z_1, z_2, z_3 , and z_4 . For an input $\vec{a} = (a_1, a_2, \dots, a_n)$, and a hash function $h \in \mathcal{H}$, define the estimator $B_h = (\sum_i h(a_i))^2$. Note that this can be computed in one pass. It is well-known that $\mathbb{E}_h[B_h] = F_2(\vec{a})$ and that $\text{Var}_h[B_h] \leq 2(\mathbb{E}_h[B_h])^2 = 2F_2(\vec{a})^2$. Let the aggregator SUM2 denote square of the sum of the input values.

THEOREM 6. *For a probabilistic stream (\vec{a}, \vec{p}) , the quantity $\text{SUM2}(h(\vec{a}), \vec{p})$ is an unbiased estimator of $F_2(\vec{a}, \vec{p})$, if h is chosen uniformly from a 4-wise independent hash family on $[n]$ with range $\{-1, +1\}$. The estimator can be computed in one-pass and yields a one-pass probabilistic stream algorithm for computing F_2 to within a relative error of ε with high probability, using space and update time of $O(\varepsilon^{-2} \log n)$*

PROOF. By the previous discussion,

$$F_2(\vec{b}) = \mathbb{E}_h[\text{SUM2}(h(\vec{b}))],$$

so the premise of Lemma 9 is satisfied with $Q = F_2$ and $Q' = \text{SUM2}$, so $F_2(\vec{a}, \vec{p}) = \mathbb{E}_h[\text{SUM2}(h(\vec{a}), \vec{p})]$. We now show that $\text{SUM2}(h(\vec{a}), \vec{p})$ can be computed efficiently over the probabilistic stream $(h(\vec{a}), \vec{p})$.

It will be convenient to introduce the jointly independent random variables U_i , for $i \in [n]$ where each U_i equals 1 with probability p_i and equals 0 otherwise. Then

$$\begin{aligned} \text{SUM2}(h(\vec{a}), \vec{p}) &= \mathbb{E}_h\left[\left(\sum_i h(a_i)U_i\right)^2\right] \\ &= \sum_i h(a_i)^2 \mathbb{E}[U_i^2] + \sum_{i \neq j} h(a_i)h(a_j) \mathbb{E}[U_i U_j] \\ &= \sum_i h(a_i)^2 p_i + \sum_{i \neq j} h(a_i)h(a_j) p_i p_j \\ &= \left(\sum_i h(a_i) p_i\right)^2 + \sum_i p_i(1 - p_i), \end{aligned}$$

using the fact that $h(a)^2 = 1$. This is easy to compute in one pass.

Finally, we show that this basic estimator can be used to produce a good estimate with high confidence. The approach is a standard one—first we compute the variance of the basic estimator. Let $P_a = \sum_{i:a_i=a} p_i$ for every $a \in [n]$. Then, we can write

$$\begin{aligned} \text{SUM2}(h(\vec{a}), \vec{p}) &= \left(\sum_i h(a_i) p_i\right)^2 + \sum_i p_i(1 - p_i) \\ &= \left(\sum_a h(a) P_a\right)^2 + \sum_i p_i(1 - p_i). \end{aligned}$$

Therefore,

$$\begin{aligned} \text{Var}_h[\text{SUM2}(h(\vec{a}), \vec{p})] &= \text{Var}_h\left[\left(\sum_a h(a) P_a\right)^2 + \sum_i p_i(1 - p_i)\right] \\ &= \text{Var}_h\left[\left(\sum_a h(a) P_a\right)^2\right], \end{aligned} \tag{2}$$

since the second summation is a constant. This expression is very similar to the one encountered in the variance bound of F_2 , except that the values P_a are generalized frequencies that can be non-integral. On the other hand, the same derivation shows that

$$\begin{aligned} \text{Var}_h\left[\left(\sum_a h(a) P_a\right)^2\right] &\leq 2\mathbb{E}_h\left[\left(\sum_a h(a) P_a\right)^2\right]^2 \\ &\leq 2\mathbb{E}_h\left[\left(\sum_a h(a) P_a\right)^2 + \sum_i p_i(1 - p_i)\right]^2 \\ &= 2\mathbb{E}_h[\text{SUM2}(h(\vec{a}), \vec{p})]^2 \end{aligned} \tag{3}$$

Combining Equations 2 and 3 shows that the variance of the estimator is at most twice the square of its expectation. Therefore, the standard technique of taking the average of $O(1/\varepsilon^2)$ such estimators reduces the relative error to ε . As usual the confidence can be increased by taking the median of sufficiently many estimators; we omit the standard calculations. \square

6. COMBINING RELATED OLAP QUERIES

Consider the problem of computing aggregators for probabilistic data of related OLAP queries involving *roll-ups* and *drill-downs*, where it is desirable to use the results computed

in previous queries. The following is a well-known strategy that can be used to speed up the computations: given an aggregator A , and a set of probabilistic data $\{S_1, S_2, \dots, S_k\}$, what are the *sufficient statistics* that one needs to compute on each S_i so that it is possible to compute A on the composite stream S_1, S_2, \dots, S_k ? For example, if A equals **MEAN**, then one solution is to compute **SUM** and **COUNT** for each stream. This is a valid approach only when the data is not probabilistic (which is not surprising since otherwise it would yield a simple data stream algorithm for **MEAN**). The sufficient statistics for **SUM** and **COUNT** are just their values on the individual probabilistic streams.

For **MEAN**, observe that it only requires maintaining the values of P_k and A_k for $k = 1, 2, \dots, O(\log(1/\varepsilon))$. (For convenience, we assume here that $P \geq 6 \ln(2/\varepsilon)$.) To combine one stream with sufficient statistics P_k, A_k with another having P'_k, A'_k , we simply add the corresponding statistics: $P_k + P'_k, A_k + A'_k$. From this, the estimate for **MEAN** can be generated. Thus the number of sufficient statistics needed for any stream is just $O(\log \varepsilon^{-1})$.

For F_2 , the sufficient statistics are just $\sum_i p_i(1 - p_i)$ together with $\sum_i h(a_i) p_i$ for each of the chosen hash functions. (The hash functions chosen must be the same across all streams.) Again, to combine streams, we simply add their corresponding sufficient statistics.

For **MIN**, the algorithm in [20] divides the domain into a sequence of $O(1/\varepsilon)$ geometrically increasing intervals (bins) and iteratively computes two quantities for each bin: (1) the probability, over possible worlds, that the minimum lies within the bin, and (2) the probability, over possible worlds, that the minimum does not belong to previous bins. The key aspect of that algorithm is that for each item in the input, all that is required of the item is the knowledge of those same two quantities. Consequently, the sufficient statistics for each stream are just these values computed for all bins. The same principle also applies to **MAX**; we omit the details.

Although this approach seems amenable to a variety of sketching algorithms, it is not clear how to combine queries using different aggregators, e.g. estimating the **MIN** of the **MEAN** of different streams. We leave it as an open problem.

7. CONCLUDING REMARKS

A number of remarkable algorithmic ideas have been developed for estimating aggregates over deterministic streams since the seminal work of [1]. Some of them are applicable to estimating aggregates over probabilistic streams such as when estimating **MEDIAN** and **REPEAT-RATE** by suitable reductions, but for other aggregates such as **DISTINCT** and **MEAN**, we need new ideas that we have presented here. The probabilistic stream model was introduced in [20] mainly motivated by probabilistic databases where data items have a distribution associated with them because of the uncertainties and inconsistencies in the data sources. This model has other applications too, including in the motivating scenario we described here in which the stream (topic distribution of search queries) derived from the deterministic input stream (search terms) is probabilistic. We believe that the probabilistic stream model will be very useful in practice in dealing with such applications.

There are several technical and conceptual open problems in addition to the question of combining queries using different aggregators mentioned in the previous section. For example, could one characterize problems for which there

is a (deterministic or randomized) reduction from probabilistic streams to deterministic streams without significant loss in space bounds or approximations? We suspect that for additive approximations, there is a simple characterization. Also, can we extend the solutions for estimating the basic aggregates we have presented here to others, in particular, geometric aggregates [19] or aggregate properties of graphs [13, 14]?

8. REFERENCES

- [1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
- [2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002.
- [3] M. Balazinska, H. Balakrishnan, and M. Stonebraker. Load management and high availability in the medusa distributed stream processing system. In *ACM SIGMOD International Conference on Management of Data*, pages 929–930, 2004.
- [4] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10, 2002.
- [5] T. Batu, S. Kannan, S. Khanna, and A. McGregor. Reconstructing strings from random traces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 910–918, 2004.
- [6] D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. In *International Conference on Very Large Data Bases (VLDB)*, pages 970–981, 2005.
- [7] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. In *International Conference on Very Large Data Bases (VLDB)*, pages 970–981. VLDB Endowment, 2005.
- [8] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Efficient allocation algorithms for OLAP over imprecise data. In *International Conference on Very Large Data Bases (VLDB)*, pages 391–402, 2006.
- [9] K. L. Chang and R. Kannan. The space complexity of pass-efficient algorithms for clustering. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1157–1166, 2006.
- [10] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *ACM SIGMOD International Conference on Management of Data*, 2007.
- [11] C. D. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.
- [12] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *International Conference on Very Large Data Bases (VLDB)*, pages 864–875, 2004.
- [13] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: the value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.
- [14] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
- [15] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.
- [16] N. Fuhr and T. Roelleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.
- [17] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD International Conference on Management of Data*, pages 58–66, 2001.
- [18] S. Guha and A. McGregor. Space-Efficient Sampling. In *AISTATS*, pages 169–176, 2007.
- [19] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *ACM Symposium on Theory of Computing*, pages 373–380, 2004.
- [20] T. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.
- [21] S. Kannan and A. McGregor. More on reconstructing strings from random traces: Insertions and deletions. In *IEEE International Symposium on Information Theory*, pages 297–301, 2005.
- [22] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.
- [23] S. Muthukrishnan. Data streams: Algorithms and applications. *Now Publishers*, 2006.