Bottom-k and Priority Sampling, Set Similarity and Subset Sums with Minimal Independence

Mikkel Thorup
AT&T Labs—Research and University of Copenhagen
mikkel2thorup@gmail.com

ABSTRACT

We consider bottom-k sampling for a set X, picking a sample $S_k(X)$ consisting of the k elements that are smallest according to a given hash function h. With this sample we can estimate the frequency $f = |Y|/|X|$ of any subset Y as $|S_k(X) \cap Y|/k$. A standard application is the estimation of the Jaccard similarity $f = |A \cap B|/|A \cup B|$ between sets A and B. Given the bottom-k samples from A and B, we construct the bottom-k sample of their union as $S_k(A \cup B) = S_k(S_k(A) \cup S_k(B))$, and then the similarity is estimated as $|S_k(A \cup B) \cap S_k(A) \cap S_k(B)|/k$.

We show here that even if the hash function is only 2-independent, the expected relative error is $O(1/\sqrt{fk})$. For $fk = \Omega(1)$ this is within a constant factor of the expected relative error with truly random hashing.

For comparison, consider the classic approach of repeated min-wise hashing, where we use k independent hash functions $h_1, ..., h_k$, storing the smallest element with each hash function. For min-wise hashing, there can be a constant bias with constant independence, and this is not reduced with more repetitions k. Recently Feigenblat et al. showed that bottom-k circumvents the bias if the hash function is 8-independent and k is sufficiently large. We get down to 2-independence for any k. Our result is based on a simply union bound, transferring generic concentration bounds for the hashing scheme to the bottom-k sample, e.g., getting stronger probability error bounds with higher independence.

For weighted sets, we consider priority sampling which adapts efficiently to the concrete input weights, e.g., benefiting strongly from heavy-tailed input. This time, the analysis is much more involved, but again we show that generic concentration bounds can be applied.

Categories and Subject Descriptors

E.1 [**Data**]: Data Structures; G.1.2 [**Discrete Mathematics**]: Probability and Statistics

General Terms

Algorithms, Measurement, Performance, Reliability, Theory

Keywords

Sampling, Independence, Estimation

1. INTRODUCTION

The concept of min-wise hashing (or the "MinHash algorithm" according to [1]) is a basic algorithmic tool suggested by Broder et al. [6, 8] for problems related to set similarity and containment. After the initial application of this algorithm in the early Altavista search engine to detecting and clustering similar documents, the scheme has reappeared in numerous other applications[1] and is now a standard tool in data mining where it is used for estimating similarity [6,8,9], rarity [13], document duplicate detection [7,20,21,36], etc [2,4,10,29].

In an abstract mathematical view, we have two sets, A and B, and we are interested in understanding their overlap in the sense of the Jaccard similarity $f = \frac{|A \cap B|}{|A \cup B|}$. In order to do this by sampling, we need sampling correlated between the two sets, so we sample by hashing. Consider a hash function $h : A \cup B \to [0, 1]$. For now we assume that h is fully random, and has enough precision that collisions are avoided with probability 1. The main mathematical observation is that $\Pr[\text{argmin} h(A) = \text{argmin} h(B)]$ is precisely $f = |A \cap B| \,/\, |A \cup B|$. Thus, we may sample the element with the minimal hash from each set, and use them in $[\text{argmin}\, h(A) = \text{argmin}\, h(B)]$ for an unbiased estimate of f. Here, for a logical statement S, $[S] = 1$ if S is true; otherwise $[S] = 0$.

For more concentrated estimators, we use repetition with k independent hash functions, $h_1, ..., h_k$. For each set A, we store $M^k(A) = (\text{argmin}\, h_1(A), ..., \text{argmin}\, h_k(A))$, which is a sample with replacement from A. The Jaccard similarity between sets A and B is now estimated as $|M^k(A) \cap M^k(B)|/k$ where $|M^k(A) \cap M^k(B)|$ denotes the number of agreeing coordinates between $M^k(A)$ and $M^k(B)$. We shall refer to this approach as repeated min-wise or $k\times$*min*.

For our discussion, we consider the very related application where we wish to store a sample of a set X that we can use to estimate the frequency $f = \frac{|Y|}{|X|}$ of any subset $Y \subseteq X$. The idea is that the subset Y is not known when the sample from X is made. The subset Y is revealed later in the form of a characteristic function that can tell if (sampled) elements belong to Y. Using the $k\times$min sample $M^k(X)$, we

[1]See `http://en.wikipedia.org/wiki/MinHash`

estimate the frequency as $|M^k(X) \cap Y|/k$ where $|M^k(X) \cap Y|$ denotes the number of samples from $M^k(X)$ in Y.

Another classic approach for frequency estimation is to use just one hash function h and use the k elements from X with the smallest hashes as a sample $S_k(X)$. This is a sample without replacement from X. As in [12], we refer to this as a bottom-k sample. The method goes back at least to [19]. The frequency of Y in X is estimated as $|Y \cap S_k(X)|/k$. Even though surprisingly fast methods have been proposed to compute $k \times \min$ [3], the bottom-k signature is much simpler and faster to compute. In a single pass through a set, we only apply a single hash function h to each element, and use a max-priority queue to maintain the k smallest elements with respect to h.

It is standard[1] to use bottom-k samples to estimate the Jaccard similarity between sets A and B, for this is exactly the frequency of the intersection in the union. First we construct the bottom-k sample $S_k(A \cup B) = S_k(S_k(A) \cup S_k(B))$ of the union by picking the k elements from $S_k(A) \cup S_k(B)$ with the smallest hashes. Next we return $|S_k(A) \cap S_k(B) \cap S_k(A \cup B)|/k$.

Stepping back, for subset frequency, we generally assume that we can identify samples from the subset. In the application to set similarity, it important that the samples are coordinated via hash functions, for this is what allows us to identify samples from the intersection as being sampled in both sets. In our mathematical analysis we will focus on the simpler case of subset frequency estimation, but it the application to set similarity that motivates our special interest in sampling via hash functions.

Limited independence.

The two approaches $k \times \min$ and bottom-k are similar in spirit, starting from the same base $1 \times \min = \text{bottom-}1$. With truly random hash functions, they have essentially the same relative standard deviation (standard deviation divided by expectation) bounded by $1/\sqrt{fk}$ where f is the set similarity or subset frequency. The two approaches are, however, very different from the perspective of pseudo-random hash functions of limited independence [35]: a random hash function h is d-independent if the hash values of any d given elements are totally random.

With min-wise hashing, we have a problem with bias in the sense of sets in which some elements have a better than average chance of getting the smallest hash value. It is known that $1 + o(1)$ bias requires $\omega(1)$-independence [26]. This bias is not reduced by repetitions as in $k \times \min$. However, recently Porat et al. [18] proved that the bias for bottom-k vanishes for large enough $k \gg 1$ if we use 8-independent hashing. Essentially they get an expected relative error of $O(1/\sqrt{fk})$, and error includes bias. For $fk = \Omega(1)$, this is only a constant factor worse than with truly random hashing. Their results are cast in a new framework of "d-k-min-wise hashing", and the translation to our context is not immediate.

Results.

In this paper, we prove that bottom-k sampling preserves the expected relative error of $O(1/\sqrt{fk})$ with 2-independent hashing, and this holds for any k including $k = 1$. We note that when $fk = o(1)$, then $1/\sqrt{fk} = \omega(1)$, so our result does not contradict a possible large bias for $k = 1$.

We remark that we also get an $O(1/\sqrt{(1-f)k})$ bound on the expected relative error. This is important if we estimate the dissimilarity $1 - f$ of sets with large similarity $f = 1 - o(1)$.

For the more general case of weighted sets, we consider priority sampling [17] which adapts near-optimally to the concrete input weights [31], e.g., benefiting strongly from heavy-tailed input. We show that 2-independent hashing suffices for good confidence intervals.

Our positive finding with 2-independence contrasts recent negative results on the insufficiency of low independence, e.g., that linear probing needs the 5-independence [26] that was proved sufficient by Pagh et al. [25].

Implementation.

For 2-independent hashing we can use the fast multiplication-shift scheme from [14], e.g., if the elements are 32-bit keys, we pick two random 64-bit numbers a and b. The hash of key x is computed with the C-code $(a * x + b) >> 32$, where $*$ is 64-bit multiplication which discards overflow, and $>>$ is a right shift. This is 10-20 times faster than the fastest known 8-independent hashing based on a degree 7 polynomial tuned for a Mersenne prime field [34][2].

Practical relevance.

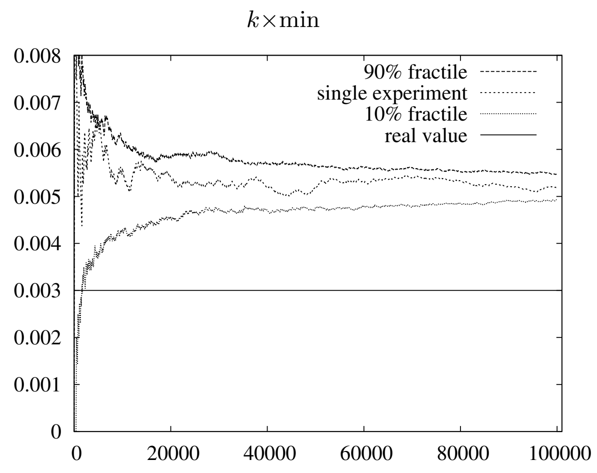
We note that Mitzenmacher and Vadhan [22] have proved that 2-independence generally works if the input has enough entropy. However, the real world has lots of low entropy data. In [34] it was noted how consecutive numbers with zero entropy made linear probing with 2-independent hashing extremely unreliable. This was a problem in connection with denial-of-service attacks using consecutive IP-addresses. For our set similarity, we would have similar issues in scenarios where small numbers are more common, hence where set intersections are likely to be fairly dense intervals of small numbers whereas the difference is more likely to consists of large random outliers. Figure 1 presents an experiment showing what happens if we try to estimating the dissimilarity with 2-independent hashing.

Stepping back, the result Mitzenmacher and Vadhan is that 2-independence works for sufficiently random input. In particular, we do not expect problems to show up in random tests. However, this does not imply that 2-independent hashing can be trusted on real data unless we have special reasons to believe that the input has high entropy. In Figure 1, bottom-k performs beautifully with 2-independent hashing, but no amount of experiments can demonstrate general reliability. However, the mathematical result of this paper is that bottom-k can indeed be trusted with 2-independent hashing: the expected relative error is $O(1/\sqrt{fk})$ no matter the structure of the input.

Techniques.

To appreciate our analysis, let us first consider the trivial case where we are given a non-random threshold probability p and sample all elements that hash below p. As in [16] we refer to this as *threshold sampling*. Since the hash of a element x is uniform in $[0, 1]$, this samples x with probability p. The sampling of x depends only on the hash value of x, so if, say, the hash function is d-independent, then the number

[2]See Table 2 in [34] for comparisons with different key lengths and computers between multiplication-shift (TwoIndep), and tuned polynomial hashing (CWtrick). The table considers polynomials of degree 3 and 4, but the cost is linear in the degree, so the cost for degree 7 is easily extrapolated.



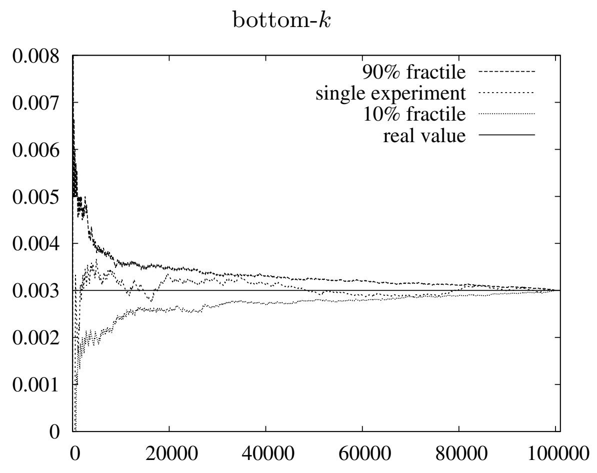


Figure 1: Experiment with set consisting of 100300 32-bit keys. It has a "core" consisting of the consecutive numbers $1, \ldots 100000$. In addition it has 300 random "outliers". Using k samples from the whole set, we want to estimate the frequency of the outliers. The true frequency is $\frac{300}{100300} \approx 0.003$. We used $k = 1, \ldots, 100000$ in $k\times$min and bottom-k and made one hundred experiments. For each k, we sorted the estimates, plotting the 10th and 90th value, labeled as 10% and 90% fractile in the figures. We also plotted the results from a single experiment. For readability, only one in every 100 values of k is plotted. Both schemes converge, but due to bias, $k\times$min converges to a value that is 70% too large. Since bottom-k does sampling without replacement, it becomes exact when the number of samples is the size of the whole set. The bias is a function of the structure of the subset within the whole set, e.g., the core set must have a negative bias complimenting the positive bias of the outliers. It is therefore not possible to correct for the bias if one only has the sample available.

of samples is the sum of d-independent 0-1 variables. This scenario is very well understood (see, e.g., [15, 30]).

We could set $p = k/n$, and get an expected number of k samples. Morally, this should be similar to a bottom-k sample, which is what we get if we end up with exactly k samples, that is, if we end up with $h_{(k)} < p \leq h_{(k+1)}$ where $h_{(i)}$ denotes the ith smallest hash value. What complicates the situation is that $h_{(k)}$ and $h_{(k+1)}$ are random variables depending on all the random hash values.

An issue with threshold sampling is that the number of samples is variable. This is an issue if we have bounded capacity to store the samples. With k expected samples, we could put some limit $K \gg k$ on the number of samples, but any such limit introduces dependencies that have to be understood. Also, if we have room for K samples, then it would seem wasteful not fill it with a full bottom-K sample.

Our analysis of bottom-k samples is much simpler than the one in [18] for 8-independent hashing with $k \gg 1$. With a union bound we reduce the analysis of bottom-k samples to the trivial case of threshold sampling. Essentially we only get a constant loss in the error probabilities. With 2-independent hashing, we then apply Chebyshev's inequality to show that the expected relative error is $O(1/\sqrt{fk})$. The error probability bounds are immediately improved if we use hash functions with higher independence.

It is already known from [5] that we can use a 2-independent bottom-k sample of a set to estimate its size n with an expected error of $O(\sqrt{n})$. The estimate is simply the inverse of the kth smallest sample. Applying this to two $\Theta(n)$-sized sets and their union, we can estimate $|A|$, $|B|$, $|A \cup B|$ and $|A \cap B| = |A| + |B| - |A \cup B|$ each with an expected error of $O(\sqrt{n})$. However, $|A \cap B|$ may be much smaller than $O(\sqrt{n})$. If we instead multiply our estimate of the similarity $f = |A \cap B|/|A \cup B|$ with the estimate of $|A \cup B|$, the resulting estimate of $|A \cap B|$ is

$$(1 \pm O(1/\sqrt{fk}) f(|A \cup B| \pm O(\sqrt{n})) = |A \cap B| \pm O(\sqrt{|A \cap B}).$$

The analysis of priority sampling for weighted sets is much more delicate, but again, using union bounds, we show that generic concentration bounds apply.

2. BOTTOM-K SAMPLES

We are given a set of X of n elements. A hash function maps the elements uniformly and collision free into $[0, 1]$. Our bottom-k sample S consists of the k elements with the lowest hash values. The sample is used to estimate the frequency $f = |Y|/|X|$ of any subset Y of X as $|Y \cap S|/k$. With 2-independent hashing, we will prove the following error probability bound for any $r \leq \bar{r} = \sqrt{k}/3$:

$$\Pr\left[||Y \cap S| - fk| > r\sqrt{fk}\right] \leq 4/r^2. \tag{1}$$

The result is obtained via a simple union bound where stronger hash functions yield better error probabilities. With d-independence with d an even constant, the probability bound is $O(1/r^d)$.

It is instructive to compare d-independence with the idea of storing d independent bottom-k samples, each based on 2-independence, and use the median estimate. Generally, if the probability of a certain deviation is p, the deviation probability for the median is bounded by $(2ep)^{d/2}$, so the $4/r^2$ from (1) becomes $(2e4/r^2)^{d/2} < (5/r)^d$, which is the same type of probability that we get with a single d-independent hash function. The big advantage of a single d-independent hash function is that we only have to store a single bottom-k sample.

If we are willing to use much more space for the hash function, then we can use twisted tabulation hashing [27] which is very fast, and then we get exponential decay in r though only down to an arbitrary polynomial the space used.

In order to show that the expected relative error is $O(1/\sqrt{fk})$, we also prove the following bound for $fk \leq 1/4$:

$$\Pr[|Y \cap S| \geq \ell] = O(fk/\ell^2 + \sqrt{f}/\ell). \tag{2}$$

From (1) and (2) we get

Proposition 1. *For bottom-k samples based on 2-independent hashing, a fraction f subset is estimated with an expected relative error of $O(1/\sqrt{fk})$.*

Proof. The proof assumes (1) and (2). For the case $fk > 1/4$, we will apply (1). The statement is equivalent to saying that the sample error $||Y \cap S| - fk|$ in expectation is bounded by $O(\sqrt{fk})$. This follows immediately from (1) for errors below $\bar{r}\sqrt{fk} = k\sqrt{|Y|/n}$. However, by (1), the probability of a larger error is bounded by $4/\bar{r}^2 = O(1/k)$. The maximal error is k, so the contribution to the expected error is $O(1)$. This is $O(\sqrt{fk})$ since $fk > 1/4$.

We will now handle the case $fk \leq 1/4$ using (2). We want to show that the expected absolute error is $O(\sqrt{fk})$. We note that only positive errors can be bigger than fk, so if the expected error is above $2fk$, the expected number of samples from Y is proportional to the expected error. We have $\sqrt{fk} \geq 2fk$, so for the expected error bound, it suffices to prove that the expected number of samples is $|Y \cap S| = O(\sqrt{fk})$. Using (2) for the probabilities, we now sum the contributions over exponentially increasing sample sizes.

$$\begin{aligned}
\mathsf{E}[|Y \cap S|] &\leq \sum_{i=0}^{\lfloor \lg k \rfloor} \left(2^{i+1} \Pr[|Y \cap S| \geq 2^i]\right) \\
&= \sum_{i=0}^{\lfloor \lg k \rfloor} O\left(2^i (fk/2^{2i} + \sqrt{f}/2^i)\right) \\
&= O\left(fk + \sqrt{f}(1 + \lg k)\right) = O\left(\sqrt{fk}\right).
\end{aligned}$$

□

2.1 A union upper bound

First we consider overestimates. For positive parameters a and b to be chosen, we will bound the probability of the overestimate

$$|Y \cap S| > \frac{1+b}{1-a} fk. \tag{3}$$

Define the threshold probability

$$p = \frac{k}{n(1-a)}.$$

Note that p is defined deterministically, independent of any samples. It is easy to see that the overestimate (3) implies one of the following two threshold sampling events:

(A) The number of elements from X that hash below p is less than k. We expected $pn = k/(1-a)$ elements, so k is a factor $(1-a)$ below the expectation.

(B) Y gets more than $(1+b)p|Y|$ hashes below p, that is, a factor $(1+b)$ above the expectation.

To see this, assume that both (A) and (B) are false. When (A) is false, we have k hashes from X below p, so the largest hash in S is below p. Now if (B) is also false, we have at most $(1+b)p|Y| = (1+b)/(1-a) \cdot fk$ elements from Y hashing below p, and only these elements from Y could be in S. This contradicts (3). By the union bound, we have proved

Proposition 2. *The probability of the overestimate (3) is bounded by $P_A + P_B$ where P_A and P_B are the probabilities of the events (A) and (B), respectively.*

Upper bound with 2-independence.

Addressing events like (A) and (B), let m be the number of elements in the set Z considered, e.g., $Z = X$ or $Z = Y$. We study the number of elements hashing below a given threshold $p \in [0,1]$. Assuming that the hash values are uniform in $[0,1]$, the mean is $\mu = mp$. Assuming 2-independence of the hash values, the variance is $mp(1-p) = (1-p)\mu$ and the standard deviation is $\sigma = \sqrt{(1-p)\mu}$. By Chebyshev's inequality, we know that the probability of a deviation by $r\sigma$ is bounded by $1/r^2$. Below we will only use that the relative standard deviation σ bounded by $1/\sqrt{\mu}$.

For any given $r \leq \sqrt{k}/3$, we will fix a and b to give a combined error probability of $2/r^2$. More precisely, we will fix $a = r/\sqrt{k}$ and $b = r/\sqrt{fk}$. This also fixes $p = k/(n(1-a))$. We note for later that $a \leq 1/3$ and $a \leq b$. This implies

$$(1+b)/(1-a) \leq (1+3b) = 1 + 3r/\sqrt{fk}. \tag{4}$$

In connection with (A) we study the number of elements from X hashing below p. The mean is $pn \geq k$ so the relative standard deviation is less than $1/\sqrt{k}$. It follows that a relative error of $a = r/\sqrt{k}$ corresponds to at least r standard deviations, so

$$P_A = \Pr\left[\#\{x \in X | h(x) < p\} < (1-a)np\right] < 1/r^2.$$

In connection with (B) we study the number of elements from Y hashing below p. Let $m = |Y|$. The mean is $pm = km/(n(1-a))$ and the relative standard deviation less than $1/\sqrt{pm} < 1/\sqrt{km/n}$. It follows than a relative error of $b = r/\sqrt{km/n}$ is more than r standard deviations, so

$$P_B = \Pr\left[\#\{y \in Y | h(y) < p\} > (1+b)mp\right] < 1/r^2.$$

By Proposition 2 we conclude that the probability of (3) is bounded by $2/r^2$. Rewriting (3) with (4), we conclude that

$$\Pr\left[|Y \cap S| > fk + 3r\sqrt{fk}\right] \leq 2/r^2. \tag{5}$$

This bounds the probability of the positive error in (1). The above constants 3 and 2 are moderate, and they can easily be improved if we look at asymptotics. Suppose we want good estimates for subsets Y of frequency at least $f_{\min}$, that is, $|Y| \geq f_{\min}|X|$. This time, we set $a = r/\sqrt{fk}$, and then we get $P_A \leq f/r^2$. We also set $b = r/\sqrt{fk}$ preserving $P_B \leq 1/r^2$. Now for any $Y \subseteq X$ with $|Y| > fn$, we have

$$\begin{aligned}
&\Pr\left[|Y \cap S| > (1+\varepsilon)fk\right] = (1+f)/r^2 \qquad (6)\\
&\quad \text{where } \varepsilon = \frac{1 + r/\sqrt{fk}}{1 - r/\sqrt{k}} - 1 = \frac{r/\sqrt{k} + r/\sqrt{fk}}{1 - r/\sqrt{k}}.
\end{aligned}$$

With $f = o(1)$ and $k = \omega(1)$, the error is $\varepsilon = (1 + o(1))r/\sqrt{fk}$, and the error probability is $P_\varepsilon = (1+f)/r^2 = (1+o(1))/r^2$. Conversely, this means that if we for subsets of frequency f and a relative positive error ε want an error probability around P_ε, then we set $r = \sqrt{1/P_\varepsilon}$ and $k = r^2/(f\varepsilon^2) = 1/(f\, P_\varepsilon\, \varepsilon^2)$.

2.2 A union lower bound

We have symmetric bounds for underestimates:

$$|Y \cap S| < \frac{1-b'}{1+a'} fk. \tag{7}$$

This time we define the threshold probability $p' = \frac{k}{n(1+a')}$. It is easy to see that the overestimate (3) implies one of the following two events:

(A′) The number of elements from X below p' is at least k. We expected $p'n = k/(1+a')$ elements, so k is a factor $(1+a')$ above the expectation.

(B′) Y gets less than $(1-b')p|Y|$ hashes below p', that is, a factor $(1-b')$ below the expectation.

To see this, assume that both (A′) and (B′) are false. When (A′) is false, we have less than k hashes from X below p', so S must contain all hashes below p'. Now if (B) is also false, we have at least $(1-b)p'|Y| = (1-b)/(1+a) \cdot fk$ elements from $Y \subseteq X$ hashing below p', hence which must be in S. This contradicts (7). By the union bound, we have proved

Proposition 3. *The probability of the underestimate (7) is bounded by $P_{A'} + P_{B'}$ where $P_{A'}$ and $P_{B'}$ are the probabilities of the events (A′) and (B′), respectively.*

Lower bound with 2-independence.

Using Proposition 3 we will bound the probability of underestimates, complementing our previous probability bounds for overestimates from Section 2.1. We will provide bounds for the same overall relative error as we did for the overestimates; namely

$$\varepsilon = \frac{1+b}{1-a} - 1 = (a+b)/(1-a)$$

However, for the events (A′) and (B′) we are going to scale up the relative errors by a factor $(1+a)$, that is, we will use $a' = a(1+a)$ and $b' = b(1+a)$. The overall relative negative error from (7) is then

$$\begin{aligned}\varepsilon' &= 1 - \frac{1-b'}{1+a'} = (a'+b')/(1+a') \\ &< (1+a)(a+b)/(1+a') < (a+b) < \varepsilon.\end{aligned}$$

Even with this smaller error, we will get better probability bounds than those we obtained for the overestimates. For (A) we used $1/\sqrt{k}$ as an upper bound on the relative standard deviation, so a relative error of a was counted as $s_A = a\sqrt{k}$ standard deviations. In (A′) we have mean $\mu' = np' = k/(1+a')$, so the relative standard deviation is bounded by $1/\sqrt{k/(1+a')} = \sqrt{1+a+a^2}/\sqrt{k}$. This means that for (A′), we can count a relative error of $a' = a(1+a)$ as

$$\begin{aligned}s'_A &= a(1+a)\sqrt{k}/\sqrt{1+a+a^2} \\ &= s_A(1+a)/\sqrt{1+a+a^2} > s_A\end{aligned}$$

standard deviations. In Section 2.1 we bounded P_A by $1/s_A^2$, and now we can bound $P_{A'}$ by $1/{s'_A}^2 \leq 1/s_A^2$. The scaling has the same positive effect on our probability bounds for (B′). That is, in Section 2.1, a relative error of b was counted as $s_B = b\sqrt{fk}$ standard deviations. With (B′) our relative error of $b' = b(1+a)$ is counted as

$$\begin{aligned}s'_B &= b(1+a)\sqrt{fk}/\sqrt{1+a+a^2} \\ &= s_B(1+a)/\sqrt{1+a+a^2} > s_B\end{aligned}$$

standard deviations, and then we can bound $P_{B'}$ by $1/{s'_B}^2 \leq 1/s_B^2$. Summing up, our negative relative error ε' is smaller than our previous positive error ε, and our overall negative error probability bound $1/{s'_A}^2 + 1/{s'_B}^2$ is smaller than our previous positive error probability bound $1/{s_A}^2 + 1/{s_B}^2$. We therefore translate (5) to

$$\Pr\left[|Y \cap S| < fk - 3r\sqrt{fk}\right] \leq 2/r^2. \tag{8}$$

which together with (5) establishes (1). Likewise (6) translates to

$$\Pr\left[||Y \cap S| - fk| > \varepsilon fk)\right] \leq 2(1+f)/r^2 \tag{9}$$

$$\text{where } \varepsilon = \frac{1+r/\sqrt{fk}}{1-r/\sqrt{k}} - 1.$$

As for the positive error bounds we note that with $f = o(1)$ and $k = \omega(1)$, the error is $\varepsilon = (1+o(1))r/\sqrt{fk}$ and the error probability is $P_\varepsilon = (2+o(1))/r^2$. Conversely, this means that if we for a target relative error ε want an error probability around P_ε, then we set $r = \sqrt{2/P_\varepsilon}$ and $k = r^2/(f\varepsilon^2) = 2/(f\, P_\varepsilon\, \varepsilon^2)$.

2.3 Rare subsets

We now consider the case where the expected number fk of samples from Y is less than 1/4. We wish to prove (2)

$$\Pr[|Y \cap S| \geq \ell] = O(fk/\ell^2 + \sqrt{f}/\ell).$$

For some balancing parameter $c \geq 2$, we use the threshold probability $p = ck/n$. The error event (A) is that less than k elements from X sample below p. The error event (B) is that at least ℓ elements hash below p. As in Proposition 2, we observe that ℓ bottom-k samples from Y implies (A) or (B), hence that $\Pr[|Y \cap S| \geq \ell] \leq P_A + P_B$.

The expected number of elements from X that hash below p is ck. The error event (A) is that we get less than k, which is less than half the expectation. This amounts to at least $\sqrt{ck}/2$ standard deviations, so by Chebyshev's inequality, the probability of (A) is $P_A \leq 1/(\sqrt{ck}/2)^2 = 4/(ck)$.

The event (B) is that at least ℓ elements from Y hash below p, while the expectation is only fck. Assuming that $\ell \geq 2fck$, the error is by at least $(\ell/2)/\sqrt{fck}$ standard deviations. By Chebyshev's inequality, the probability of (B) is $P_B \leq 1/((\ell/2)/\sqrt{fck})^2 = 4fck/\ell^2$. Thus

$$P_A + P_B \leq 4/(ck) + 4fck/\ell^2.$$

We wish to pick c for balance, that is,

$$4/ck = 4fck/\ell^2 \iff c = \ell/(\sqrt{f}k)$$

However, we have assumed that $c \geq 2$ and that $\ell \geq 2fck$. The latter is satisfied because $2fck = 2fk\ell/(\sqrt{f}k) = 2\sqrt{f}\ell$ and $f \leq 1/4$. Assuming that $c = \ell/(\sqrt{f}k) \geq 2$, we get

$$P_A + P_B \leq 8/(k(\ell/(\sqrt{f}k))) = 8\sqrt{f}/\ell.$$

When $\ell/(\sqrt{f}k) < 2$, we set $c = 2$. Then

$$P_A + P_B \leq 2/k + 8fk/\ell^2 \leq 16fk/\ell^2.$$

Again we need to verify that $\ell \geq fck = 2fk$, but that follows because $\ell \geq 1$ and $fk \leq 1/4$. We know that at one of the above two cases applies, so we conclude that

$$P[|Y \cap S| \geq \ell] \leq P_A + P_B = O(fk/\ell^2 + \sqrt{f}/\ell),$$

completing the proof of (2).

3. PRIORITY SAMPLING

We now consider the more general situation where we are dealing with a set I of weighted items with w_i denoting the weight of item $i \in I$. Let $\sum I = \sum_{i \in I} w_i$ denote the total weight of set I.

Now that we are dealing with weighted items, we will use *priority sampling* [17] which generalizes the bottom-k samples we used for unweighted elements. The input is a set of I of weighted items. Each item or element i is identified by a unique key which is hashed uniformly to a random number $h_i \in (0,1)$. The item is assigned a *priority* $q_i = w_i/h_i > w_i$. We assume that all priorities end up distinct and different from the weights. If not, we could break ties based on an ordering of the items. The priority sample S of size k contains the k samples of highest priority, but it also stores a threshold τ which is the $(k+1)$th highest priority. Based on this we assign a weight estimate $\widehat{w}_i$ to each item i. If i is not sampled, $\widehat{w}_i = 0$; otherwise $\widehat{w}_i = \max\{w_i, \tau\}$. A basic result from [17] is that $\mathsf{E}[\widehat{w}_i] = w_i$ if the hash function is truly random (in [17], the h_i were described as random numbers, but here they are hashes of the keys).

We note that priority sampling generalize the bottom-k sample we used for unweighted items, for if all weights are unit, then the k highest priorities correspond to the k smallest hash values. In fact, priority sampling predates [12], and [12] describes bottom-k samples for weighted items as a generalization of priority sampling, picking the first k items according to an arbitrary randomized function of the weights.

The original objective of priority sampling [17] was subset sum estimation. A subset $J \subseteq I$ of the items is selected, and we estimate the total weight in the subset as $\widehat{w}_J = \sum\{\widehat{w}_i | i \in J \cap S\}$. By linearity of expectation, this is an unbiased estimator. A cool application from [17] was that as soon as the signature of the Slammer worm [23] was identified, we could inspect the priority samples from the past to track its history and identify infected hosts. An important point is that the Slammer worm was not known when the samples were made. Samples are made with no knowledge on which subsets will later turn out to be of interest.

Trivially, if we want to estimate the relative subset weight $\sum J / \sum I$ and we do not know the exact total, we can divide $\widehat{w}_J$ with the estimate $\widehat{w}_I$ of the total. As with the bottom-k sampling for unweighted items, we can easily use priority sampling to estimate the similarity of sets of weighted items: given the priority sample from two sets, we can easily construct the priority sample of their union, and estimate the intersection as a subset. This is where it is important that we use a hash function so that the sampling from different sets is coordinated, e.g., we could not use iterative sampling procedures like the one in [11]. In the case of histogram similarity, it is natural to allow the same item to have different weights in different sets. More specifically, allowing zero weights, every possible item has a weight in each set. For the similarity we take the sum of the minimum weight for each item, and divide it by the sum of the maximum weight for each item. This requires a special sampling that we shall return to at the end.

Priority sampling is not only extremely easy to implement on-line with a standard min-priority queue; it also has some powerful universal properties in its adaption to the concrete input weights. As proved in [31], given one extra sample, priority sampling has smaller variance sum $\sum_i \mathsf{Var}[\widehat{w}_i]$ than *any* off-line sampling scheme tailored for the concrete input weights. In particular, priority sampling benefits strongly if there are dominant weights w_i in the input, estimated precisely as $\widehat{w}_i = \max\{w_i, \tau\} = w_i$. In the important case of heavy tailed input distributions [1], we thus expect most of the total weight to be estimated without any error. The quality of a priority sample is therefore often much better than what can be described in terms of simple parameters such as total weight, number of items, etc. The experiments in [17] on real and synthetic data show how priority sampling sometimes gains orders of magnitude in estimate quality over competing methods.

The quality of a priority estimate depends completely on the distribution of weights in the input, and often we would like to know how much we can trust a given estimate. What we really want from a sample is not just an estimate, but a confidence interval for the subset sum [32]: from the sample we want to compute lower and upper bounds that capture the true value with some desired probability. The confidence interval does not have to be a simple nice function. It has to be efficiently computable, and we want it to be as tight as possible.

All the current analysis of priority sampling [17,31,32] is heavily based on true randomness, assuming that the priorities are independent random variables, e.g., the proof from [17] that $\mathsf{E}[\widehat{w}_i] = w_i$ starts by fixing the priorities q_j of all the other items $j \neq i$. However, in this paper, we want to use hash functions with independence as low as 2, and then any such approach breaks down. Estimates will no longer be unbiased, but we will show that good confidence intervals can still be computed.

To the best of our knowledge, our paper is the first to show that anything useful can be said about subset sum estimates based on given number of k samples made using $<$ 8-independent random variables. The essence of our result is that priority sampling gets reasonable performance with any hashing scheme as long as it is 2-independent.

Below we first develop error probability bounds for priority sampling with limited independence applied to a given set of input weights. Afterwards we show how confidence intervals can be derived from a sample. At the very end, we show how to estimate histogram similarity.

3.1 Threshold sampling

Generalizing the pattern for unweighted sets, our basic goal is to relate the error probabilities with priority sampling to the much simpler case of threshold sampling for weighted items [16]. In our description of threshold sampling, we will develop notation and basic results that will later be used for priority sampling.

In threshold sampling, we do not have a predefined sample size. Instead we have a given threshold t. We will still use exactly the same random priorities as above, but now an item is sampled if and only if $q_i > t$. The weight estimate is

$$\widehat{w}_i^t = \begin{cases} 0 & \text{if } q_i \leq t \\ \max\{w_i, t\} & \text{if } q_i > t \end{cases} \tag{10}$$

Thus, if priority sampling leads to threshold τ, then the priority estimates are $\widehat{w}_i = \widehat{w}_i^\tau$. We shall use I^t to denote the items sampled with threshold t, that is, items i with $q_i > t$. The priority threshold τ is the $(k+1)$st largest priority, hence the smallest value such that $|I^\tau| \leq k$.

We note that threshold sampling is well-known from statistics (see, e.g., [28]). With fully random and independent h_i, each item i is sampled independently: a so-called a Poisson sampling scheme. With the given weights w_i and threshold t, let k_t be the expected number of samples with threshold t. Among all possible Poisson sampling schemes with unbiased estimators $\widehat{w}_i$ and an expected number of k_t samples, threshold sampling is the unique scheme that minimizes the variance sum $\sum_{i\in I} \mathsf{Var}[\widehat{w}_i]$ [28, p 86].

Fractional subsets and inner products.

It is convenient to generalize from regular subsets to fractional subsets. For each $i \in I$, there is a fraction $f_i \in [0,1]$. We want to estimate fw denoting the inner product $\sum_{i\in I} f_i w_i$. We estimate it as $f\widehat{w}^t$ denoting $\sum_{i\in I} f_i \widehat{w}_i^t = \sum_{i\in S} f_i \widehat{w}_i^t$. Note that for this estimate, we only need to know f_i for $i \in S$. To emulate a standard subset J, we let f be the characteristic function of J, that is, $f_i = 1$ if $i \in J$; otherwise $f_i = 0$ otherwise. Using inner products will simplify a lot of notation in our analysis. The generalization to fractional subsets comes for free in our analysis which is all based on concentration bounds for sums of random variables $X_i \in [0,1]$.

Notation for smaller weights.

Whenever we with threshold or priority sampling end up with a threshold t, we know that variability in the estimate is from items i with weight below t. We will generally use a subscript $_{<t}$ to denote the restriction to items i with weights $w_i < t$, e.g., $I_{<t} = \{i \in I | w_i < t\}$ and $w_{<t} = (w_i)_{i\in I_{<t}}$ is the vector of these smaller weights. Then $fw_{<t} = \sum_{i\in I_{<t}} f_i w_i$. Notice that $fw_{<t}$ does not include i with $w_i \geq t$ even if $f_i w_i < t$.

Above we defined $w_{<t}$ to denote the vector $(w_i)_{i\in I_{<t}}$ of weights below t, and used it for the inner product $fw_{<t} = \sum_{i\in I_{<t}} f_i w_i$. When it is clear from the context that we need a number, not a vector, we will use $w_{<t}$ to denote the sum of these weights, that is, $w_{<t} = \sum_{i\in I_{<t}} w_i = \underline{1} w_{<t}$ where $\underline{1}$ is the all 1s vector. Since $f_i \leq 1$ for all i, we always have $fw_{<t} \leq w_{<t}$.

We shall use subscript $_{\leq t}$, $_{\geq t}$, and $_{>t}$ to denote the corresponding restriction to items with weight $\leq t$, $\geq t$, and $> t$, respectively.

For a subset J of I, we let $fJ = \sum_{i\in J} f_i$, thus identifying J with its characteristic vector. As an example, we can write our estimate with threshold t as

$$f\widehat{w}^t = fw_{\geq t} + t(fI_{<t}^t). \tag{11}$$

Error probability functions.

As in the unweighted case, the point in relating to threshold sampling is that error probability bounds for threshold sampling are easily derived. For items with unit weights, we reduced the bottom-k sampling error event to the union of four threshold sampling error events (A), (B), (A'), and (B'). However, now with weighted items, we are going to reduce the priority sampling error event to the union of an unbounded number of threshold sampling error events that happen with geometrically decreasing probabilities. Our reduction will hold for most hash functions, including 2-independent hash functions, but to make such a claim clear, we have to carefully describe what properties of the hash functions we rely on.

Assume that the threshold t is fixed. With reference to (11), the variability in our estimate is all from $fI_{<t}^t$. An item $i \in I_{<t}$ is sampled and included in $I_{<t}^t$ if $q_i = w_i/h_i > t \iff h_i < w_i/t$, hence with probability w_i/t. If i is sampled, it adds $f_i \in [0,1]$ to $fI_{<t}^t$; otherwise 0.

The above is an instance of bounded random variables $X_i \in [0,1]$, $i \in I_{<t}$, where each X_i is a function of h_i; namely $X_i = f_i[h_i < w_i/t]$. With the fixed threshold t, X_i depends only on the hash h_i. Therefore, if the hash values h_i are, say d-independent, then so are the X_i. Let $X = \sum_{i\in I_{<t}} X_i$ and $\mu = \mathsf{E}[X]$. We are interested in an error probability function $\wp$ such that for every $\mu > 0$, $\delta > 0$, if $\mu = \mathsf{E}[X]$, then

$$\Pr[|X - \mu| > \delta\mu] \leq \wp(\mu, \delta).$$

The error probability function $\wp$ that we can use depends on the quality of the hash function. For example, if the hash function is 2-independent, then $\mathsf{Var}[X] \leq \mu$, and then by Chebyshev's inequality, we can use

$$\wp(\mu, \delta) = 1/(\delta^2\mu). \tag{12}$$

For most of our results, it is more natural to think of δ as a function of μ and some target error probability $P \in (0,1)$, defining $\delta(\mu, P)$ such that

$$\mu(\mu, \delta(\mu, P)) = P. \tag{13}$$

Returning to threshold sampling with threshold t, for $i \in I_{<t}$, we have $X_i = f_i[i \in I_{<t}^t]$, $X = fI_{<t}^t$, and $\mu = fw_{<t}/t$. Moreover, by (11), $f\widehat{w}^t - fw = t(fI_{<t}^t) - fw_{<t}$. Hence

$$\Pr[|f\widehat{w}^t - fw| > \delta(fw_{<t}/t, P) fw_{<t}] \leq P. \tag{14}$$

When we start analyzing priority sampling, we will need to relate the probabilities of different threshold sampling events. This places some constraints on the error probability function $\wp$. Mathematically, it is convenient to allow $\wp$ to attain values above 1, but only the values below 1 are probabilistically intersting.

Definition 4. *An error probability function $\wp : \mathbb{R}_{\geq 0} \times R_{\geq 0} \to R_{\geq 0}$ is* well-behaved *if*

(a) $\wp$ is continuous and strictly decreasing in both arguments.

(b) If with the same absolute error we decrease the expectancy, then the probability goes down. Formally if $\mu' < \mu$ and $\mu'\delta' \geq \mu\delta$, then $\wp(\mu', \delta') < \wp(\mu, \delta)$.

(c) If $\delta \leq 1$ and $\wp(\mu, \delta) < 1$, then $\wp(\mu, \delta)$ falls at least inversely proportional to $\mu\delta^2$. Formally, if $\delta_0, \delta_1 \leq 1$, $\wp(\mu_0, \delta_0) < 1$, and $\mu_0\delta_0^2 < \mu_1\delta_1^2$, then

$$\wp(\mu_0, \delta_0) \geq \frac{\mu_0\delta_0^2}{\mu_1\delta_1^2} \wp(\mu_1, \delta_1). \tag{15}$$

The conditions are trivially satisfied with our Chebyshev bound $\wp(\mu, \delta) = 1/(\mu\delta^2)$. We will use (c) to argue that probabilities of different events fall geometrically. The somewhat cryptic formulation in (c) is relevant in connection with higher independence, e.g., for full randomness we have the standard Chernoff bounds (see, e.g., [24]) that for $\delta \leq 1$

give $\wp(\mu,\delta) = 2\exp(-\delta^2\mu/3)$. This $\wp$ only satisfies (15) if $\mu_0\delta_0^2 \geq 3$. Obviously the Chernoff bounds give much faster decrease as $\mu\delta^2$ increases. The point in our results is that we only require the decrease in (15).

As an application of (a) and (b) we get

LEMMA 5. *For thresholds t', t, and relative errors δ', δ, if $t' < t$ and $\wp(fw_{<t'}/t', \delta_{<t'}) = \wp(fw_{\leq t}/t, \delta_{\leq t})$, then*

$$\delta' fw_{<t'} < \delta fw_{<t}.$$

Hence, for any fixed target error probability P in (14), the target error

$$\delta(fw_{<t}/t, P) fw_{<t}$$

is strictly decreasing in the threshold t.

PROOF. We will divide the decrease from t to t' into a series of atomic decreases. The first atomic "decrease" is from $fw_{\leq t}$ to $fw_{<t}$. This makes no difference unless there are weights equal to t so that $fw_{<t} < fw_{\leq t}$. Assume this is the case and suppose $\wp(fw_{<t}/t, \delta_{<t}) = \wp(fw_{\leq t}/t, \delta_{\leq t})$. Since $fw_{\leq t}/t < fw_{<t}/t$, it follows directly from (b) that $\delta_{<t} fw_{<t}/t < \delta_{\leq t} w_{\leq t}/t$, hence that $\delta_{<t} fw_{<t} < \delta_{\leq t} w_{\leq t}$.

The other atomic decrease we consider is from $fw_{<t}$ to $fw_{\leq t'}$ where $t' < t$ and with no weights in (t', t), hence with $fw_{\leq t'} = fw_{<t}$. Suppose $\wp(fw_{\leq t'}/t', \delta_{\leq t'}) = \wp(fw_{<t}/t, \delta_{<t})$. Since $t' < t$, $fw_{\leq t'}/t' > fw_{<t}/t$, so by (a), $\delta_{\leq t'} < \delta_{<t'}$. It follows that $\delta_{\leq t'} fw_{\leq t'} < \delta_{<t} fw_{<t}$. Alternating between these two atomic decreases, we can implement an arbitrary decrease in the threshold as required for the lemma. $\square$

Threshold confidence intervals.

In the case of threshold sampling with a fixed threshold t, it is now trivial to derive confidence intervals for the true value fw. The sample gives us the exact value $fw_{\geq t}$ for weights at least as big as t, and an estimate $f\widehat{w}_{<t}^t$ for those below. Setting

$$f\widehat{w}_{<t}^- = \min\{x \mid (1 + \delta(x/t, P))x \geq f\widehat{w}_{\geq t}^t\}$$
$$f\widehat{w}_{<t}^+ = \max\{x \mid (1 - \delta(x/t, P))x \leq f\widehat{w}_{\geq t}^t\}$$

we get

$$\Pr\left[fw_{\geq t} + f\widehat{w}_{<t}^- \;\leq\; fw \;\leq\; fw_{\geq t} + f\widehat{w}_{<t}^+\right] \geq 1 - 2P.$$

3.2 Priority sampling: the main result

We are now ready to present our main technical result.

THEOREM 6. *For items $i \in I$, let be given a weight vector $(w_i)_{i\in I}$ with corresponding fractions $(f_i)_{i\in I}$. With a given target error probability P and sample size k, consider a random priority sampling event, assigning to each item $i \in I$, a hash $h_i \in (0,1)$ and priority $q_i = w_i/h_i$. Let τ be the resulting priority threshold, i.e, the $k+1$th largest priority. Let*

$$\delta = 6\,\delta(f\widehat{w}_{<\tau}/(3\tau), P).$$

If $\delta < 2$, then

$$\Pr[|f\widehat{w}^\tau - fw| > \delta fw_{<\tau}] = O(P).$$

The above constants are not optimized, but with pure O-notation, it is not as easy to make a formally clear statement. Ignoring the constants and the restriction $\delta < 2$, we see that our error bound for priority sampling with threshold τ is of the same type as the one in (14) for threshold sampling with fixed threshold $t = \tau$. In our case, the priority threshold τ is variable, and from Lemma 5 it follows that our probabilistic error bound

$$6\,\delta(f\widehat{w}_{<\tau}/(3\tau), P)\, fw_{<\tau}$$

decreases with the priority threshold τ.

The proof of Theorem 6 is rather convoluted. With some target error probability P, we will identify $t_{\min}$, $t_{\max}$ such that

(i) With probability $1 - O(P)$, the priority threshold $\tau \in [t_{\min}, t_{\max}]$.

(ii) With probability $1-O(P)$, with a single random choice of the h_i but simultaneously for all $t \in [t_{\min}, t_{\max}]$, if $\delta = 6\,\delta(f\widehat{w}_{<t}/(3t), P) < 2$, then $|f\widehat{w}^\tau - fw| \leq \delta fw_{<t}$.

A union bound on $\neg$(i)$\vee\neg$(ii) implies Theorem 6. What makes (ii) very tricky to prove is that $\delta(f\widehat{w}_{<t}/(3t), P)$ can vary a lot for different $t \in [t_{\min}, t_{\max}]$.

Priority confidence intervals.

The format of Theorem 6 makes it very easy to derive confidence intervals like those for threshold sampling. A priority sample with priority threshold τ gives us the exact value $fw_{\geq\tau}$ for weights at least as big as τ, and an estimate $f\widehat{w}_{<\tau}^\tau$ for those below. For an upper bound on $fw_{<\tau}$, we compute

$$f\widehat{w}_{<\tau}^+ = \max\{x \mid \delta = 6\,\delta(x/(3\tau), P)) \wedge (1-\delta)x \leq f\widehat{w}_{\geq\tau}^\tau\}.$$

Note that here in the upper bound, we only consider $\delta \leq 1$, so we do not need to worry about the restriction $\delta < 2$ in Theorem 6. For the lower bound, we use

$$f\widehat{w}_{<\tau}^- = \min\{x \mid \delta = 6\,\delta(x/(3\tau), P)) < 2 \wedge (1+\delta)x \geq f\widehat{w}_{\geq\tau}^\tau\}.$$

Here in the lower bound, the restriction $\delta = 6\,\delta(x/(3\tau), P)) < 2$ prevents us from deriving a lower bound $x = f\widehat{w}_{<\tau}^- \leq f\widehat{w}_{\geq\tau}^\tau/3$. In such cases, we use the trivial lower bound $x = f\widehat{w}_{<\tau}^- = 0$ which in distance from $f\widehat{w}_{<\tau}^\tau$ is at most 1.5 times bigger. Now, by Theorem 6,

$$\Pr\left[fw_{\geq\tau} + f\widehat{w}_{<\tau}^- \;\leq\; fw \;\leq\; fw_{\geq\tau} + f\widehat{w}_{<\tau}^+\right] \geq 1 - O(P).$$

In cases where the exact part $fw_{\geq\tau}$ of an estimate is small compared with the variable part $f\widehat{w}_{\geq\tau}^\tau$, we may be interested in a non-zero lower bound $f\widehat{w}_{<\tau}^-$ even if it is smaller than $f\widehat{w}_{\geq\tau}^\tau/3$. To do this, we need bounds for larger δ.

Large errors.

We are now going to present bounds that works for arbitrarily large relative errors δ. The bounds are not as clean as those from Theorem 6, but we include them to show that something can be done also for $\delta \geq 2$. In particular, this means that we only worry about positive errors.

THEOREM 7. *For items $i \in I$, let be given a weight vector $(w_i)_{i\in I}$ with corresponding fractions $(f_i)_{i\in I}$, a target error probability P, and a sample size k. Based on $(w_i)_{i\in I}$, let $t_{\max}$ the smallest upper bound on a random priority threshold that is exceeded with probability at most P, that is, the probability of generating at least $k+1$ priorities above $t_{\max}$ is at most P. Consider a random priority sampling event. Set*

$$\ell = 1 + \log_2(t_{\max}/\tau)$$
$$\delta = \delta(fw_{<\tau}/\tau,\; P/\ell_\tau^2).$$

Then

$$\Pr[f\widehat{w}^{\tau}_{<\tau} > (2+2\delta_\tau)fw_{<\tau}] = O(P)$$

Complementing Theorem 6, we only intend to use Theorem 7 for large errors where $(1+2\delta_\tau) = O(\delta_\tau)$. We wish to provide a probabilistic lower bound for $fw_{<\tau}$. Unfortunately, we do not know $t_{\max}$ which depends on the whole weight vector $(w_i)_{i\in I}$. However, based our priority sample, it is not hard to generate a probabilistic upper bound $\bar{t}_{\max}$ on $t_{\max}$ such that $\Pr[\bar{t}_{\max} < t_{\max}] \le P$. We then compute $\bar{\ell}_\tau = 1 + \log_2(\bar{t}_{\max}/\tau)$ and set

$$f\widehat{w}^{-}_{<\tau} = \min\{x \,|\, \bar{\delta} = \delta(x/\tau, P/\bar{\ell}^2)) \wedge 2(1+\bar{\delta})x \ge f\widehat{w}^{\tau}_{<\tau}\}.$$

Then by Theorem 7,

$$\Pr\left[fw_{<\tau} \;\ge\; f\widehat{w}^{-}_{<\tau}\right] = 1 - O(P).$$

To see this, let $f\widehat{w}^{*}_{<\tau}$ be the value we would have obtained if we had computed $f\widehat{w}^{-}_{<\tau}$ using the real $t_{\max}$. Our error event is that $\bar{t}_{\max} < t_{\max}$ or $fw_{<\tau} \;<\; f\widehat{w}^{*}_{<\tau}$. The former happens with probability at most P, and Theorem 7 states that the latter happens with probability $O(P)$. Hence none of these error events happen with probability $1-O(P)$, but then $\bar{t}_{\max} \ge t_{\max}$, implying $f\widehat{w}^{-}_{<\tau} \le f\widehat{w}^{*}_{<\tau} \le fw_{<\tau}$.

For space reasons, the proofs of Theorems 6 and 7 are deferred to the full version [33] of this paper.

3.3 Histogram similarity

As mentioned earlier, with weighted items, given the priority samples of sets A and B, we can easily estimate the weight of their intersection and union. As for bottom-k sample, the point is that we can construct the priority sample of their union. This priority sample involves the top $k+1$ priorities from $A \cup B$. If one of these is in A, then it must also be among the top $k+1$ priorities from A.

However, if we want to compare histograms, then it is natural to say that the same item i may have different weights in different sets. The item i has weight w_i^A in A and weight w_i^B in B. Let $w_i^{\max} = \max\{w_i^A, w_i^B\}$ and $w_i^{\min} = \min\{w_i^A, w_i^B\}$. The histogram similarity is $w^{\min}/w^{\max} = (\sum_i w_i^{\min})/(\sum_i w_i^{\max})$.

This would seem a perfect application of our fractional subsets with $w_i = w_i^{\max}$ and $f_i = w_i^{\min}/w_i^{\max}$. The issue is as follows. From our priority samples over the w_i^A and w_i^B we can easily get the priority sample for the $w_i = w_i^{\max}$. However, for the i sampled, we would typically not have a sample with $w_i^{\min}$, and then we cannot compute f_i.

Our solution is to keep the instances of an item i in A and B separate as twins i^A and i^B with priorities $q_i^A = w_i^A/h_i$ and $q_i^B = w_i^B/h_i$. For coordination, we still use the same hash value h_i to determine the priorities. If $w_i^A = w_i^B$, we get $q_i^A = q_i^B$, and then we break the tie in favor of i^A. The priority sample for $A \cup B$ consists of the top k split items, and the priority threshold τ is the $k+1$ biggest among all priorities. Estimation is done as usual. The important point here is the interpretation of the results. If $w_i^A \ge w_i^B$, then the priority of i^A is higher than that of i^B. Thus, in our sample, when we see an item i^C, $C \in \{A, B\}$, we count it for the union $\widehat{w}^{\max}$ if it is not preceded by its twin; otherwise we count it for the intersection $\widehat{w}^{\min}$.

The resulting estimators $\widehat{w}^{\min}$ and $\widehat{w}^{\max}$ will no longer be unbiased with truly random hashing. However, for our probabilistic error bounds with pseudo-random hashing, there is no asymptotic effect to our analysis. All the current analysis is using union bounds over threshold sampling events, using the fact that each hash value h_i contributes at most 1 to the number of items with priorities above a given threshold t. Now h_i affects 2 twins, but this is fine since all we need is that the contribution of each random variable is bounded by a constant.

4. REFERENCES

[1] R. Adler, R. Feldman, and M. Taqqu. *A Practical Guide to Heavy Tails*. Birkhauser, 1998.

[2] Y. Bachrach, R. Herbrich, and E. Porat. Sketching algorithms for approximating rank correlations in collaborative filtering systems. In *Proc. 16th SPIRE*, pages 344–352, 2009.

[3] Y. Bachrach and E. Porat. Fast pseudo-random fingerprints. *CoRR*, abs/1009.5791, 2010.

[4] Y. Bachrach, E. Porat, and J. S. Rosenschein. Sketching techniques for collaborative filtering. In *Proc. 21st IJCAI*, pages 2016–2021, 2009.

[5] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 1–10, 2002.

[6] A. Z. Broder. On the resemblance and containment of documents. In Proc. Compression and Complexity of Sequences (SEQUENCES), pages 21–29, 1997.

[7] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proc. 11th CPM*, pages 1–10, 2000.

[8] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000. See also STOC'98.

[9] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29:1157–1166, 1997.

[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.

[11] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Efficient stream sampling for variance-optimal estimation of subset sums. *SIAM Journal on Computing*, 40(5):1402–1431, 2011. Announced at SODA'09.

[12] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *Proc. 26th PODC*, pages 225–234, 2007.

[13] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proc. 10th ESA*, pages 323–334, 2002.

[14] M. Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proc. 13th STACS*, pages 569–580, 1996.

[15] M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable (extended abstract). In *Proc. 19th ICALP*, pages 235–246, 1992.

[16] N. Duffield, C. Lund, and M. Thorup. Learn more, sample less: control of volume and variance in network measurements. *IEEE Transactions on Information Theory*, 51(5):1756–1775, 2005.

[17] N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54(6):Article 32, 2007. Announced at SIGMETRICS'04.

[18] G. Feigenblat, E. Porat, and A. Shiftan. Even better framework for min-wise based algorithms. *CoRR*, abs/1102.3537, 2011. Accepted as "Exponential space improvement for min-wise based algorithms" for FSTTCS'12.

[19] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proc. SIGMOD Conference*, pages 171–182, 1997.

[20] M. R. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proc. 29th SIGIR*, pages 284–291, 2006.

[21] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *Proc. 16th WWW*, pages 141–150, 2007.

[22] M. Mitzenmacher and S. P. Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *Proc. 19th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 746–755, 2008.

[23] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security and Privacy Magazine*, 1(4):33–39, 2003.

[24] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[25] A. Pagh, R. Pagh, and M. Ružić. Linear probing with constant independence. *SIAM Journal on Computing*, 39(3):1107–1120, 2009. See also STOC'07.

[26] M. Pătraşcu and M. Thorup. On the k-independence required by linear probing and minwise independence. In *Proc. 36th ICALP, Part I, LNCS 6198*, pages 715–726, 2010.

[27] M. Pătraşcu and M. Thorup. Twisted tabulation hashing. In *Proc. 23nd SODA*, pages 209–228, 2013.

[28] C.-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling.* Springer, 1992.

[29] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proc. SIGMOD*, pages 76–85, 2003.

[30] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995. See also SODA'93.

[31] M. Szegedy. The DLT priority sampling is essentially optimal. In *Proc. 38th STOC*, pages 150–158, 2006.

[32] M. Thorup. Confidence intervals for priority sampling. In *Proc. SIGMETRICS*, pages 252–263, 2006.

[33] M. Thorup. Bottom-k and priority sampling, set similarity and subset sums with minimal independence. *CoRR*, 2013.

[34] M. Thorup and Y. Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012. Announced at SODA'04 and ALENEX'10.

[35] M. N. Wegman and L. Carter. New classes and applications of hash functions. *Journal of Computer and System Sciences*, 22(3):265–279, 1981. See also FOCS'79.

[36] H. Yang and J. P. Callan. Near-duplicate detection by instance-level constrained clustering. In *Proc. 29th SIGIR*, pages 421–428, 2006.