New cardinality estimation algorithms for HyperLogLog sketches

Otmar Ertl
otmar.ertl@gmail.com

December 26, 2016

This paper presents new methods to estimate the cardinalities of multisets recorded by HyperLogLog sketches. A theoretically motivated extension to the original estimator is presented that eliminates the bias for small and large cardinalities. Based on the maximum likelihood principle a second unbiased method is derived together with a robust and efficient numerical algorithm to calculate the estimate. The maximum likelihood approach can also be applied to more than a single HyperLogLog sketch. In particular, it is shown that it gives more accurate cardinality estimates for the union, intersection, or relative complements of two sets that are represented by HyperLogLog sketches compared to conventional techniques like the inclusion-exclusion principle. All the new methods are demonstrated and verified by extensive simulations.

1. Introduction

Counting the number of distinct elements in a data stream or large datasets is a common problem in big data processing. Often there are parallel streams or data is spread over a cluster, which makes this task even more challenging. In principle, finding the number of distinct elements n with a maximum relative error ε in a data stream requires $\Omega(n)$ space [1]. However, probabilistic algorithms that achieve the requested accuracy only with high probability are able to drastically reduce space requirements. Many different probabilistic algorithms have been developed over the past two decades [2, 3]. An algorithm with an optimal space complexity of $\Omega(\varepsilon^{-2} + \log n)$ [1, 4] was finally presented [5]. This algorithm, however, is not very efficient in practice [3].

This paper together with source code for all presented algorithms and simulations is available at `https://github.com/oertl/hyperloglog-sketch-estimation-paper`. The first version of this paper was published on April 17, 2016.

Currently, the most memory efficient algorithm that also works for distributed setups is the near-optimal HyperLogLog algorithm [6] with space complexity $\Omega(\varepsilon^{-2} \log\log n + \log n)$. The originally proposed estimation method has some problems to guarantee the same estimation error over the entire range of cardinalities. It was proposed to correct the estimate by empirical means [7, 8, 9].

In case the data is not distributed and results do not need to be aggregated further, there are even more efficient estimation algorithms available. On the one hand there is the self-learning bitmap [10], and on the other hand there is the HyperLogLog algorithm extended by a historic inverse probability estimator that is continuously updated together with the HyperLogLog sketch [3, 11]. Both achieve the same estimation error using less space. However, the estimated cardinality depends on the insertion order of elements and hence cannot be used in a distributed environment.

2. HyperLogLog data structure

The HyperLogLog algorithm uses a sketching data structure that consists of m registers. For performance reasons the number of registers m is chosen to be a power of 2, $m = 2^p$. p is the precision parameter that directly controls the relative error which scales like $1/\sqrt{m}$. All registers start with zero initial value. Each element insertion potentially increases the value of one of these registers. The maximum value a register can reach is a natural bound given either by the output size of the used hash algorithm or the space that is reserved for a single register. Common implementations allocate up to 8 bits per register.

2.1. Data element insertion

In order to insert a data element into a HyperLogLog data structure a hash value is calculated. The leading p bits of the hash value are used to select one of the 2^p registers. Among the next following q bits, the position of the first 1-bit is determined which is a value in the range $[1, q+1]$. The value $q+1$ is used, if all q bits are zeros. If the position of the first 1-bit exceeds the current value of the selected register, the register value is replaced. Algorithm 1 shows the update procedure for inserting a data element into the HyperLogLog sketch.

Algorithm 1 Insertion of a data element D into a HyperLogLog data structure that consists of $m = 2^p$ registers. All registers $\boldsymbol{K} = (K_1, \ldots, K_m)$ have zero initial value and remain in the range $[0, q+1]$.

procedure InsertElement(D, $\boldsymbol{K}$)
  $\langle a_1, \ldots, a_p, b_1, \ldots, b_q \rangle_2 \leftarrow (p+q)$-bit hash value of D ▷ $a_i, b_i \in \{0, 1\}$
  $i \leftarrow 1 + \langle a_1, \ldots, a_p \rangle_2$ ▷ $i \in [1, 2^p]$
  $k \leftarrow \min(\{s \mid s \in [1, q] \wedge b_s = 1\} \cup \{q+1\})$ ▷ $k \in [1, q+1]$
  $K_i \leftarrow \max(K_i, k)$
end procedure

The desribed element insertion algorithm makes use of what is known as stochastic averaging [12]. Instead of updating each of all m registers using m independent hash values, which would be an $\mathcal{O}(m)$ operation, only one register is selected and updated, which requires only a single hash function and reduces the complexity to $\mathcal{O}(1)$.

A HyperLogLog sketch can be characterized by a parameter pair (p, q). The precision parameter p controls the relative error while the second parameter defines the possible value range of a register. A register can take all values starting from 0 to $q+1$, inclusively. The sum $p + q$ corresponds to the number of consumed hash value bits and defines the maximum cardinality that can be tracked. Obviously, if the cardinality reaches values in the order of 2^{p+q}, hash collisions will become more apparent and the estimation accuracy will be drastically reduced.

Algorithm 1 has some properties which are especially useful for distributed data streams. First, the insertion order of elements has no influence on the final HyperLogLog sketch. Furthermore, any two HyperLogLog sketches with same parameters (p, q) representing sets A and B can be easily merged. The HyperLogLog sketch that represents the union of both sets is simply constructed by taking the register-wise maximum values

$$K_i^{A\cup B} = \max\left(K_i^A, K_i^B\right) \quad \text{for } 1 \leq i \leq m. \tag{1}$$

At any time a (p, q)-HyperLogLog sketch can be compressed into a (p', q')-HyperLogLog data structure, if $p' \leq p$ and $p' + q' \leq p + q$ is satisfied (see Algorithm 2). This transformation is lossless in a sense that the resulting HyperLogLog sketch is the same as if all elements would have been recorded by a (p', q')-HyperLogLog sketch right from the beginning.

A $(p, 0)$-HyperLogLog sketch corresponds to a bit array as used by linear counting [13]. Each register value can be stored by a single bit in this case. Hence, linear counting can be regarded as a special case of the HyperLogLog algorithm for which $q = 0$.

2.2. Joint probability distribution of register values

Under the assumption of a uniform hash function, the probability that the register values $\boldsymbol{K} = (K_1, \ldots, K_m)$ of a HyperLogLog sketch with parameters p and q are equal to $\boldsymbol{k} = (k_1, \ldots, k_m)$ is given by the corresponding probability mass function

$$\rho(\boldsymbol{k}|n) = \sum_{n_1+\ldots+n_m=n} \binom{n}{n_1, \ldots, n_m} \frac{1}{m^n} \prod_{i=1}^{m} \gamma(k_i|n_i) \tag{2}$$

where n is the cardinality. The n distinct elements are distributed over all m registers according to a multinomial distribution with equal probabilities. $\gamma(k|n)$ is the probability that the value of a register is equal to k, after it was selected n times by the insertion

Algorithm 2 Compression of a (p, q)-HyperLogLog sketch with register values $\boldsymbol{K} = (K_1, \ldots, K_m)$ into a (p', q')-HyperLogLog sketch with $p' \leq p$ and $p' + q' \leq p + q$.

function COMPRESS(D, $\boldsymbol{K}$)
  allocate registers for new HyperLogLog sketch $\boldsymbol{K}' = (K_1, \ldots, K_{2^{p'}})$
  for $i \leftarrow 1, 2^{p'}$ **do**
    $b \leftarrow (i-1) \cdot 2^{p-p'}$
    $j \leftarrow 1$
    while $j \leq 2^{p-p'} \wedge K_{b+j} = 0$ **do**
      $j \leftarrow j + 1$
    end while
    if $j = 1$ **then**
      $K'_i \leftarrow \min(K_{b+j} + p - p', q' + 1)$
    else if $j \leq 2^{p-p'}$ **then**
      $\langle a_1, \ldots, a_{p-p'} \rangle_2 \leftarrow j - 1$
      $K'_i \leftarrow \min(\{s \mid s \in [1, p-p'] \wedge a_s = 1\} \cup \{q' + 1\})$
    else
      $K'_i \leftarrow 0$
    end if
  end for
  return $\boldsymbol{K}'$
end function

algorithm

$$\gamma(k|n) := \begin{cases} 1 & n = 0 \wedge k = 0 \\ 0 & n = 0 \wedge 1 \leq k \leq q+1 \\ 0 & n \geq 1 \wedge k = 0 \\ \left(1 - \frac{1}{2^k}\right)^n - \left(1 - \frac{1}{2^{k-1}}\right)^n & n \geq 1 \wedge 1 \leq k \leq q \\ 1 - \left(1 - \frac{1}{2^q}\right)^n & n \geq 1 \wedge k = q+1. \end{cases} \tag{3}$$

The order of register values $K_1, \ldots, K_m$ is not important for the estimation of the cardinality. More formally, the multiset $\{K_1, \ldots, K_m\}$ is a sufficient statistic for n. Since the values of the multiset are all in the range $[0, q+1]$ the multiset can also be represented as $\{K_1, \ldots, K_m\} = 0^{C_0} 1^{C_1} \cdots q^{C_q} (q+1)^{C_{q+1}}$ where C_k is the multiplicity of value k. As a consequence, the multiplicity vector $\boldsymbol{C} := (C_0, \ldots, C_{q+1})$ is also a sufficient statistic for the cardinality. In addition, this vector contains all the information about the HyperLogLog sketch that is required for cardinality estimation. The two HyperLogLog parameters can be obtained by $p = \log_2 \|\boldsymbol{C}\|_1$ and $q = \dim \boldsymbol{C} - 2$, respectively.

2.3. Poisson approximation

Due to the statistical dependence of the register values, the probability mass function (2) is difficult for further analysis. Therefore, a Poisson model can be used [6], which

assumes that the cardinality itself is distributed according to a Poisson distribution

$$n \sim \text{Poisson}(\lambda). \tag{4}$$

Under the Poisson model the distribution of the register values is

$$\begin{aligned}
\rho(\boldsymbol{k}|\lambda) &= \sum_{n=0}^{\infty} \rho(\boldsymbol{k}|n) e^{-\lambda} \frac{\lambda^n}{n!} && (5)\\
&= \sum_{n_1=0}^{\infty} \cdots \sum_{n_m=0}^{\infty} \prod_{i=1}^{m} \gamma(k_i|n_i) e^{-\frac{\lambda}{m}} \frac{\lambda^{n_i}}{n_i! m^{n_i}} \\
&= \prod_{i=1}^{m} \sum_{n=0}^{\infty} \gamma(k_i|n) e^{-\frac{\lambda}{m}} \frac{\lambda^n}{n! m^n} \\
&= \prod_{k=0}^{q+1} \left(\sum_{n=0}^{\infty} \gamma(k|n) e^{-\frac{\lambda}{m}} \frac{\lambda^n}{n! m^n} \right)^{c_k} \\
&= e^{-c_0 \frac{\lambda}{m}} \left(\prod_{k=1}^{q} \left(e^{-\frac{\lambda}{m 2^k}} \left(1 - e^{-\frac{\lambda}{m 2^k}} \right) \right)^{c_k} \right) \left(1 - e^{-\frac{\lambda}{m 2^q}} \right)^{c_{q+1}}. && (6)
\end{aligned}$$

Here c_k denotes the multiplicity of value k in the multiset $\{k_1, \ldots, k_m\}$. The final factorization shows that under the Poisson model the register values $K_1, \ldots, K_m$ are independent and identically distributed. The probability that a register has a value less than or equal to k for a given rate λ is defined by

$$P(K \leq k|\lambda) = \begin{cases} 0 & k < 0 \\ e^{-\frac{\lambda}{m 2^k}} & 0 \leq k \leq q \\ 1 & k > q. \end{cases} \tag{7}$$

2.4. Depoissonization

Due to the simpler probability mass function, it is easier to find an estimator $\hat{\lambda} = \hat{\lambda}(\boldsymbol{K})$ for the Poisson rate λ rather than for the cardinality n in the fixed-size model (2). Depoissonization [14] finally allows to translate the estimates back to the fixed-size model. Assume we have found an unbiased estimator for the Poisson rate

$$\mathbb{E}(\hat{\lambda}|\lambda) = \lambda \quad \text{for all } \lambda \geq 0. \tag{8}$$

We know from (5)

$$\mathbb{E}(\hat{\lambda}|\lambda) = \sum_{n=0}^{\infty} \mathbb{E}(\hat{\lambda}|n) e^{-\lambda} \frac{\lambda^n}{n!} \tag{9}$$

and therefore

$$\sum_{n=0}^{\infty} \mathbb{E}(\hat{\lambda}|n) e^{-\lambda} \frac{\lambda^n}{n!} = \lambda \tag{10}$$

holds for all $\lambda \geq 0$. The unique solution of this equation is given by

$$\mathbb{E}(\hat{\lambda}|n) = n. \tag{11}$$

Hence, the unbiased estimator $\hat{\lambda}$ conditioned on n is also an unbiased estimator for n, which motivates us to use $\hat{\lambda}$ directly as estimator for the cardinality $\hat{n} := \hat{\lambda}$. As simulation results will show later, the Poisson approximation works well over the entire cardinality range, even for estimators that are not exactly unbiased.

3. Original cardinality estimation method

The original cardinality estimator [6] is based on the idea that the number of distinct element insertions a register needs to reach the value k is proportional to $m2^k$. Given that, a rough cardinality estimate can be obtained by averaging the values $\{m2^{K_1}, \ldots, m2^{K_m}\}$.

In the history of the HyperLogLog algorithm different averaging techniques have been proposed. First, there was the LogLog algorithm using the geometric mean and the SuperLogLog algorithm that enhanced the estimate by truncating the largest register values before applying the geometric mean [15]. Finally, the harmonic mean was found to give even better estimates as it is inherently less sensitive to outliers. The result is the so-called raw estimator which is given by

$$\hat{n}_{\text{raw}} = \alpha_m \frac{m}{\frac{1}{m2^{K_1}} + \ldots + \frac{1}{m2^{K_m}}} = \frac{\alpha_m m^2}{\sum_{i=1}^{m} 2^{-K_i}} = \frac{\alpha_m m^2}{\sum_{k=0}^{q+1} C_k 2^{-k}}. \tag{12}$$

Here α_m is a bias correction factor which was derived for a given number of registers m as [6]

$$\alpha_m := \left(m \int_0^\infty \left(\log_2 \left(\frac{2+u}{1+u} \right) \right)^m du \right)^{-1}. \tag{13}$$

Numerical approximations of α_m for various values of m have been listed in [6]. These approximations are used in many HyperLogLog implementations. However, since the published constants have been rounded to 4 significant digits, these approximations even introduce some bias for very high precisions p. For HyperLogLog sketches that are used in practice with 256 or more registers ($p \geq 8$), it is completely sufficient to use

$$\alpha_\infty := \lim_{m \to \infty} \alpha_m = \frac{1}{2 \log 2} \approx 0.7213475, \tag{14}$$

as approximation for α_m in (12), because the additional bias is negligible compared to the overall estimation error.

Fig. 1 shows the distribution of the relative error for the raw estimator as function of the cardinality. The chart is based on 10 000 randomly generated HyperLogLog sketches. More details of the experimental setup will be explained later in Section 3.5. Obviously, the raw estimator is biased for small and large cardinalities where it fails to return accurate estimates. In order to cover the entire range of cardinalities, corrections for small and large cardinalities have been proposed.

As mentioned in Section 2.1, a HyperLogLog sketch with parameters (p, q) can be mapped to a $(p, 0)$-HyperLogLog sketch. Since $q = 0$ corresponds to linear counting and the reduced HyperLogLog sketch corresponds to a bitset with C_0 zeros, the linear counting cardinality estimator [13] can be used

$$\hat{n}_{\text{small}} = m \log(m/C_0). \tag{15}$$

The corresponding relative estimation error as depicted in Fig. 2 shows that this estimator is convenient for small cardinalities. It was proposed to use this estimator for small cardinalities as long as $\hat{n}_{\text{raw}} \leq \frac{5}{2}m$ where the factor $\frac{5}{2}$ was empirically determined [6].

For large cardinalities in the order of 2^{p+q}, for which a lot of registers are already in a saturated state, meaning that they have reached the maximum possible value $q + 1$, the raw estimator underestimates the cardinalities. For the 32-bit hash value case ($p + q = 32$), which was considered in [6], following correction formula was proposed to take these saturated registers into account

$$\hat{n}_{\text{large}} = -2^{32} \log\left(1 - \hat{n}_{\text{raw}}/2^{32}\right). \tag{16}$$

The original estimation algorithm as presented in [6] including corrections for small and large cardinalities is summarized by Algorithm 3. The relative estimation error for

Algorithm 3 Original cardinality estimation algorithm for HyperLogLog sketches that use 32-bit hash values ($p + q = 32$) for insertion of data items [6].

```
function EstimateCardinality(K)
    m ← dim K
    n̂_raw = α_m m² (Σ_{i=1}^{m} 2^{-K_i})^{-1}        ▷ raw estimate (12)
    if n̂_raw ≤ (5/2) m then
        C_0 = |{i | K_i = 0}|
        if C_0 ≠ 0 then
            return m log(m/C_0)                        ▷ small range correction (15)
        else
            return n̂_raw
        end if
    else if n̂_raw ≤ (1/30) 2^32 then
        return n̂_raw
    else
        return −2^32 log(1 − n̂_raw/2^32)               ▷ large range correction (16)
    end if
end function
```

the original method is shown in Fig. 3. Unfortunatley, as can be clearly seen, the ranges where the estimation error is small for $\hat{n}_{\text{raw}}$ and $\hat{n}_{\text{small}}$ do not overlap. Therefore, the estimation error is much larger near the transition region. To reduce the estimation error for cardinalities close to this region, it was proposed to correct $\hat{n}_{\text{raw}}$ for bias. Empirically collected bias correction data is either stored as set of interpolation points [7], as lookup

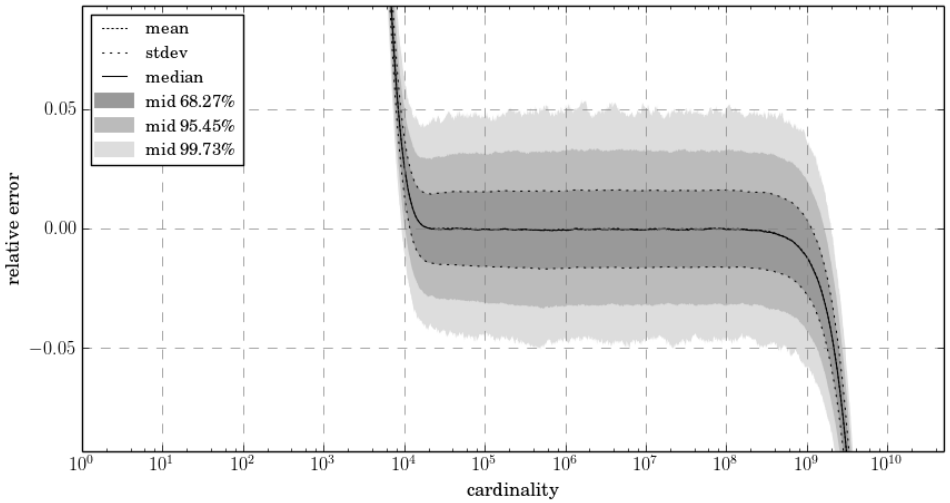

Figure 1: The distribution of the relative estimation error over the cardinality for the raw estimator after evaluation of 10 000 randomly generated HyperLogLog data structures with parameters $p = 12$ and $q = 20$.

table [8], or as best-fitting polynomial [9]. However, all these empirical approaches treat the symptom and not the cause.

The large range correction formula (16) is not satisfying either as it does not reduce the estimation error. Quite the contrary, it even makes the bias worse. However, instead of underestimating the cardinalities, they are now overestimated. Another indication for the incorrectness of the proposed large range correction is the fact that it is not even defined for all possible states. For instance, consider a (p, q)-HyperLogLog sketch with $p+q = 32$ for which all registers are equal to the maximum possible value $q+1$. The raw estimate would be $\hat{n}_{\text{raw}} = \alpha_m 2^{33}$, which is greater than 2^{32} and outside of the domain of the large range correction formula.

A simple approach to avoid the need of any large range correction is to extend the operating range of the raw estimator to larger cardinalities. This can be easily accomplished by using hash values with more bits ($p + q > 32$). If $q \geq 30$, 5-bit registers are no longer sufficient to represent all possible values in the range $[0, q+1]$. If, for example, 64-bit hash values ($p+q = 64$) are used, as proposed in [7], 6 bits per register are needed. Hash value sizes with more than 64 bits are useless in practice, because it is unrealistic to encounter cardinalities of order 2^{64} for which the raw estimator would be biased.

3.1. Derivation of the raw estimator

In order to better understand why the raw estimator fails for small and large cardinalities, we start with a brief and simple derivation without the restriction to large cardinalities ($n \to \infty$) and without using complex analysis as in [6].

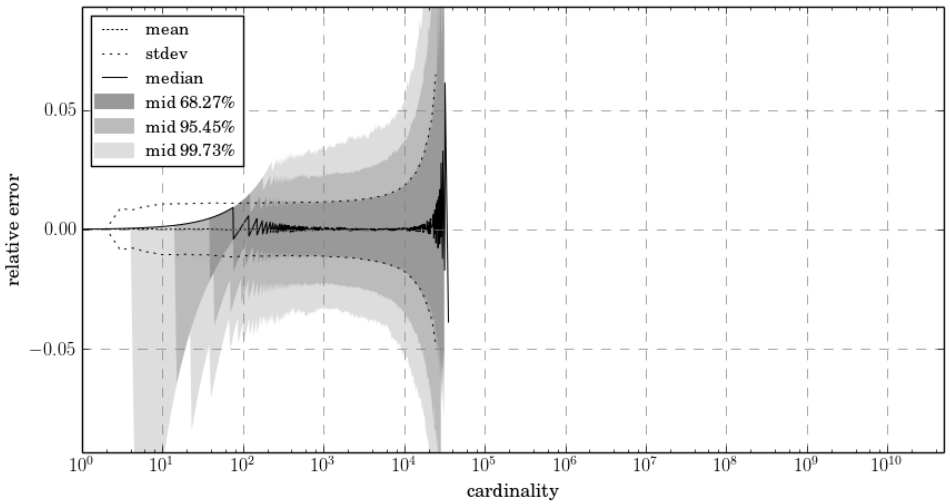

Figure 2: The distribution of the relative estimation error for the linear counting estimator after evaluation of 10 000 randomly generated bitmaps of size 2^{12} which correspond to HyperLogLog sketches with parameters $p = 12$ and $q = 0$.

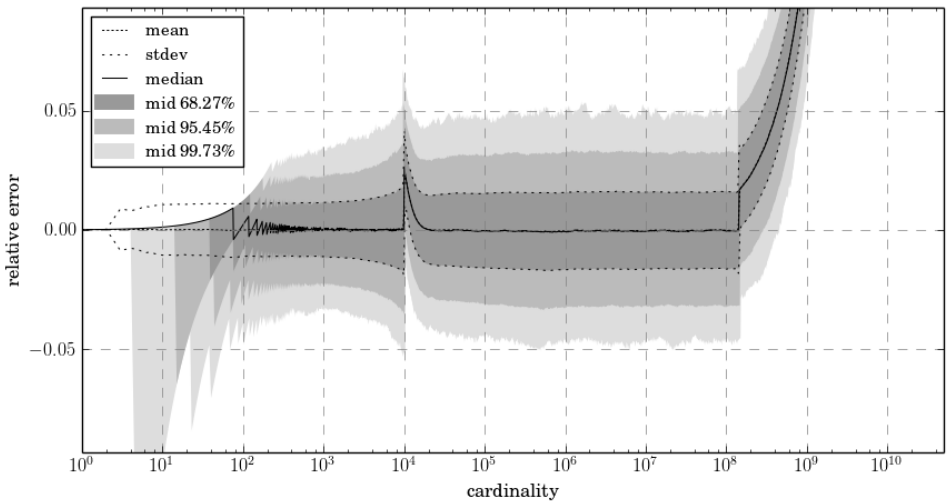

Figure 3: The distribution of the relative estimation error over the cardinality for the original estimation algorithm after evaluation of 10 000 randomly generated HyperLogLog data structures with parameters $p = 12$ and $q = 20$.

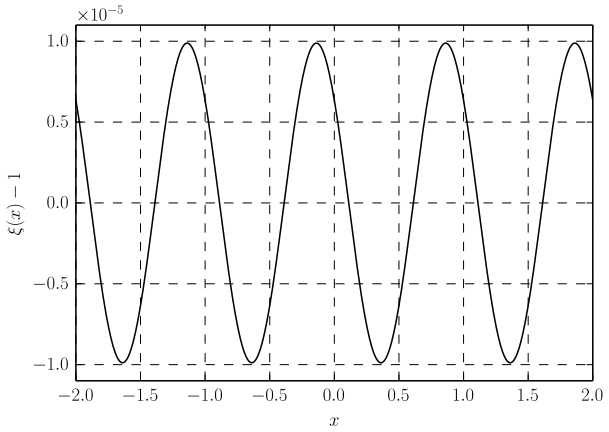

Figure 4: The deviation of $\xi(x)$ from 1.

Let us assume that the register values have following cumulative distribution function

$$P(K \leq k|\lambda) = e^{-\frac{\lambda}{m2^k}}. \tag{17}$$

For now we ignore that this distribution has infinite support and differs from the register value distribution under the Poisson model (7), whose support is limited to the range $[0, q+1]$. For a random variable K obeying (17) the expectation of 2^{-K} is given by

$$\mathbb{E}(2^{-K}) = \sum_{k=-\infty}^{\infty} 2^{-k}\left(e^{-\frac{\lambda}{m2^k}} - e^{-\frac{\lambda}{m2^{k-1}}}\right) = \frac{1}{2}\sum_{k=-\infty}^{\infty} 2^k e^{-\frac{\lambda}{m}2^k} = \frac{\alpha_\infty\, m\, \xi(\log_2(\lambda/m))}{\lambda}, \tag{18}$$

where the function

$$\xi(x) := \log(2) \sum_{k=-\infty}^{\infty} 2^{k+x} e^{-2^{k+x}} \tag{19}$$

is a smooth periodic function with period 1. Numerical evaluations indicate that this function can be bounded by $1 - \varepsilon_\xi \leq \xi(x) \leq 1 + \varepsilon_\xi$ with $\varepsilon_\xi := 9.885 \times 10^{-6}$ (see Fig. 4). This value can also be found using Fourier analysis as shown in Appendix A.

Let $K_1, \ldots, K_m$ be a sample distributed according to (17). For large sample sizes $m \to \infty$ we have asymptotically

$$\mathbb{E}\left(\frac{1}{2^{-K_1} + \ldots + 2^{-K_m}}\right) \underset{m\to\infty}{=} \frac{1}{\mathbb{E}(2^{-K_1} + \ldots + 2^{-K_m})} = \frac{1}{m\mathbb{E}(2^{-K})}. \tag{20}$$

Together with (18) we obtain

$$\lambda = \mathbb{E}\left(\frac{\alpha_\infty\, m^2\, \xi(\log_2(\lambda/m))}{2^{-K_1} + \ldots + 2^{-K_m}}\right) \quad \text{for } m \to \infty. \tag{21}$$

Therefore, the asymptotic relative bias of

$$\hat{\lambda} = \frac{\alpha_\infty \, m^2}{2^{-K_1} + \ldots + 2^{-K_m}} \tag{22}$$

is bounded by ε_ξ, which makes this statistic a good estimator for the Poisson parameter. It also corresponds to the raw estimator (12), if the Poisson parameter estimate is used as cardinality estimate (see Section 2.4).

3.2. Limitations of the raw estimator

The raw estimator is based on two prerequisites. First, the number of registers needs to be sufficiently large ($m \to \infty$). And second, the distribution of register values can be described by (17). However, the latter is not true for small and large cardinalities, which is finally the reason for the bias of the raw estimator.

A random variable K' with cumulative distribution function (17) can be transformed into a random variable K with cumulative distribution function (7) using

$$K = \min\big(\max\big(K', 0\big), q + 1\big) \, . \tag{23}$$

Therefore, register values $K_1, \ldots, K_m$ can be seen as the result after applying this transformation to a sample $K'_1, \ldots, K'_m$ of the distribution described by (17). If all registers values fall into the range $[1, q]$, they must be identical to the values $K'_1, \ldots, K'_m$. In other words, the observed register values are also a plausible sample of the assumed distribution described by (17) in this case. Hence, as long as all or at least most register values are in the range $[1, q]$, the approximation of (7) by (17) is valid. This explains why the raw estimator works best for intermediate cardinalities. However, for small and large cardinalities many register values are equal to 0 or $q+1$, respectively, which cannot be described by (17) and finally leads to the observed bias.

3.3. Corrections to the raw estimator

If we knew the values $K'_1, \ldots, K'_m$ for which transformation (23) led to the observed register values $K_1, \ldots, K_m$, we would be able to estimate λ using

$$\hat{\lambda} = \frac{\alpha_\infty \, m^2}{2^{-K'_1} + \ldots + 2^{-K'_m}}. \tag{24}$$

As we have already shown, this estimator is approximately unbiased, because all K'_i follow the assumed distribution. It would be even sufficient, if we knew the multiplicity C'_k of each $k \in \mathbb{Z}$ in $\{K'_1, \ldots, K'_m\}$, $C'_k := |\{i | k = K'_i\}|$, because the raw estimator can be also written as

$$\hat{\lambda} = \frac{\alpha_\infty \, m^2}{\sum_{k=-\infty}^{\infty} C'_k 2^{-k}}. \tag{25}$$

Due to (23), the multiplicities C'_k and the multiplicities C_k for the observed register values have following relationships

$$\begin{aligned} C_0 &= \textstyle\sum_{k=-\infty}^{0} C'_k, \\ C_k &= C'_k, \qquad\qquad 1 \le k \le q, \\ C_{q+1} &= \textstyle\sum_{k=q+1}^{\infty} C'_k. \end{aligned} \tag{26}$$

The idea is now to find estimates $\hat{c}'_k$ for all $k \in \mathbb{Z}$ and use them as replacements for C'_k in (25). For $k \in [1, q]$ where $C_k = C'_k$ we can use the trivial estimators

$$\hat{c}'_k := C_k, \quad 1 \le k \le q. \tag{27}$$

To get estimators for $k \le 0$ and $k \ge q+1$, we consider the expectation of C'_k

$$\mathbb{E}(C'_k) = m\left(P(K \le k|\lambda) - P(K \le k-1|\lambda)\right) = me^{-\frac{\lambda}{m2^k}}\left(1 - e^{-\frac{\lambda}{m2^k}}\right). \tag{28}$$

From (7) we know that $\mathbb{E}(C_0/m) = e^{-\frac{\lambda}{m}}$ and $\mathbb{E}(1 - C_{q+1}/m) = e^{-\frac{\lambda}{m2^q}}$, and therefore, we can also write

$$\mathbb{E}(C'_k) = m\left(\mathbb{E}(C_0/m)\right)^{2^{-k}}\left(1 - \left(\mathbb{E}(C_0/m)\right)^{2^{-k}}\right) \tag{29}$$

and

$$\mathbb{E}(C'_k) = m\left(\mathbb{E}(1 - C_{q+1}/m)\right)^{2^{q-k}}\left(1 - \left(\mathbb{E}(1 - C_{q+1}/m)\right)^{2^{q-k}}\right), \tag{30}$$

which motivates us to use

$$\hat{c}'_k = m\left(C_0/m\right)^{2^{-k}}\left(1 - \left(C_0/m\right)^{2^{-k}}\right) \tag{31}$$

as estimator for $k \le 0$ and

$$\hat{c}'_k = m\left(1 - C_{q+1}/m\right)^{2^{q-k}}\left(1 - \left(1 - C_{q+1}/m\right)^{2^{q-k}}\right) \tag{32}$$

as estimator for $k \ge q+1$, respectively.

Inserting all these estimators into (25) as replacements for C'_k finally gives

$$\hat{\lambda} = \frac{\alpha_\infty m^2}{\sum_{k=-\infty}^{\infty} \hat{c}'_k 2^{-k}} = \frac{\alpha_\infty m^2}{m\,\sigma(C_0/m) + \sum_{k=1}^{q} C_k 2^{-k} + m\,\tau(1 - C_{q+1}/m)2^{-q}} \tag{33}$$

which we call the improved raw estimator. Here $m\,\sigma(C_0/m)$ and $2m\,\tau(1 - C_{q+1}/m)$ are replacements for C_0 and C_{q+1} in the raw estimator (12), respectively. The functions σ and τ are defined as

$$\sigma(x) := x + \sum_{k=1}^{\infty} x^{2^k} 2^{k-1} \tag{34}$$

and

$$\tau(x) := \sum_{k=1}^{\infty} x^{2^{-k}}\left(1 - x^{2^{-k}}\right) 2^{-k}. \tag{35}$$

We can cross-check the new estimator for the linear counting case with $q = 0$. Using the identity $\sigma(x) + \tau(x) = \alpha_\infty \xi(\log_2(\log(1/x)))/\log(1/x)$, we get

$$\hat{\lambda} = \frac{\alpha_\infty m}{\sigma(C_0/m) + \tau(C_0/m)} = \frac{m \log(m/C_0)}{\xi(\log_2(\log(m/C_0)))} \tag{36}$$

which is as expected almost identical to the linear counting estimator (15), because $\xi(x) \approx 1$ (see Section 3.1).

3.4. Improved raw estimation algorithm

The improved raw estimator (33) leads to a new cardinality estimation algorithm for HyperLogLog sketches. Algorithm 4 demonstrates the numerical evaluation of the estimator. The values of functions σ and τ can be either calculated on-demand or precalculated. The possible value range of C_0 and C_{q+1} is $[0, m]$. Therefore, if performance matters, the function values can be precalculated for all possible arguments and kept in lookup tables of size $m + 1$. The new estimation algorithm is very elegant, because it does neither contain magic numbers nor special cases as the original algorithm.

3.5. Estimation error

In order to verify the new estimation algorithm, we generated 10 000 HyperLogLog sketches and inserted up to 50 billion unique elements. Assuming a uniform hash function, element hash values can be mocked by random numbers. For the following results we used the Mersenne Twister random number generator with a state size of 19 937 bits from the C++ standard library.

Fig. 5 shows the distribution of the relative error of the estimated cardinality using Algorithm 4 compared to the true cardinality for $p = 12$ and $q = 20$. As the mean shows, the error is unbiased over the entire cardinality range. The new approach is able to accurately estimate cardinalities up to 4 billions ($\approx 2^{p+q}$) which is about an order of magnitude larger than the operating range upper bound of the raw estimator (Fig. 1).

The improved raw estimator beats the precision of methods that apply bias correction on the raw estimator [7, 8, 9]. Based on the simulated data we have empirically determined the bias correction function w_{corr} for the raw estimator that satisfies $n = \mathbb{E}(w_{\text{corr}}(\hat{n}_{\text{raw}})|n)$ for all cardinalities. By definition, the estimator $\hat{n}'_{\text{raw}} := w_{\text{corr}}(\hat{n}_{\text{raw}})$ is unbiased and a function of the raw estimator. Its standard deviation can be compared with that of the improved raw estimator in Fig. 6. For cardinalities smaller than 10 000 the empirical bias correction approach is not very precise. This is the reason why all previous approaches had to switch over to the linear counting estimator at some point. The standard deviation of the linear counting estimator is also shown in Fig. 6. Obviously, the previous approaches cannot do better than given by the minimum of both curves for linear counting and raw estimator. In practice, the standard deviation is even larger, because the choice between both estimators must be made based on an estimate and not on the true cardinality, for which the intersection point of both curves represents

Algorithm 4 Cardinality estimation algorithm based on the improved raw estimator.

function EstimateCardinality($\boldsymbol{C}$)
  $m \leftarrow \|\boldsymbol{C}\|_1$
  $z \leftarrow m \cdot \tau(1 - C_{q+1}/m)$ ▷ alternatively, take $m \cdot \tau(1 - C_{q+1}/m)$ from precalculated lookup table
  for $k \leftarrow q, 1$ **do**
    $z \leftarrow 0.5 \cdot (z + C_k)$
  end for
  $z \leftarrow z + m \cdot \sigma(C_0/m)$ ▷ alternatively, take $m \cdot \sigma(C_0/m)$ from precalculated lookup table
  return $\alpha_\infty m^2/z$ ▷ $\alpha_\infty := 1/(2\log(2))$
end function

function $\sigma(x)$ ▷ $x \in [0, 1]$
  if $x = 1$ **then**
    return ∞
  end if
  $y \leftarrow 1$
  $z \leftarrow x$
  repeat
    $x \leftarrow x \cdot x$
    $z' \leftarrow z$
    $z \leftarrow z + x \cdot y$
    $y \leftarrow 2 \cdot y$
  until $z = z'$
  return z
end function

function $\tau(x)$ ▷ $x \in [0, 1]$
  $y \leftarrow 1$
  $z \leftarrow 0$
  repeat
    $x \leftarrow \sqrt{x}$
    $z' \leftarrow z$
    $y \leftarrow 0.5 \cdot y$
    $z \leftarrow z + (1 - x) \cdot x \cdot y$
  until $z = z'$
  return z
end function

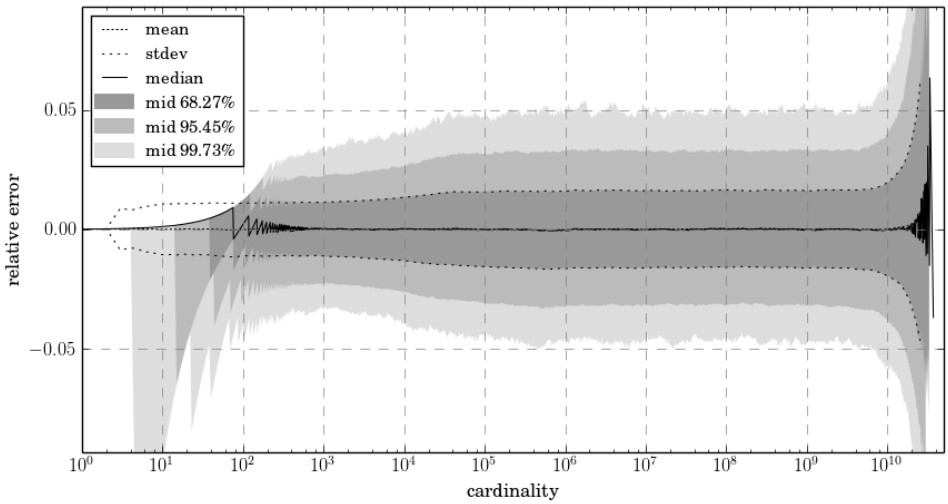

Figure 5: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 20$.

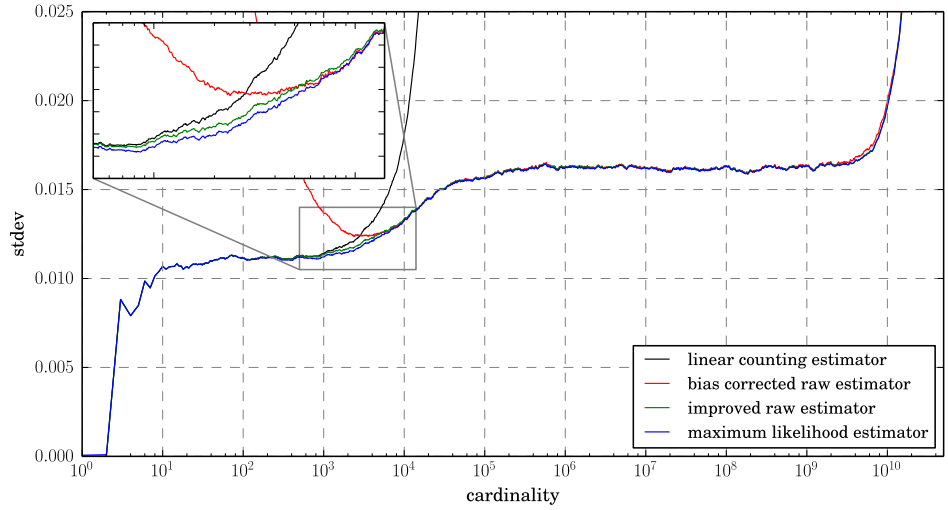

Figure 6: Standard deviations of the relative error of different cardinality estimators over the true cardinality for a HyperLogLog sketch with paramters $p = 12$ and $q = 20$.

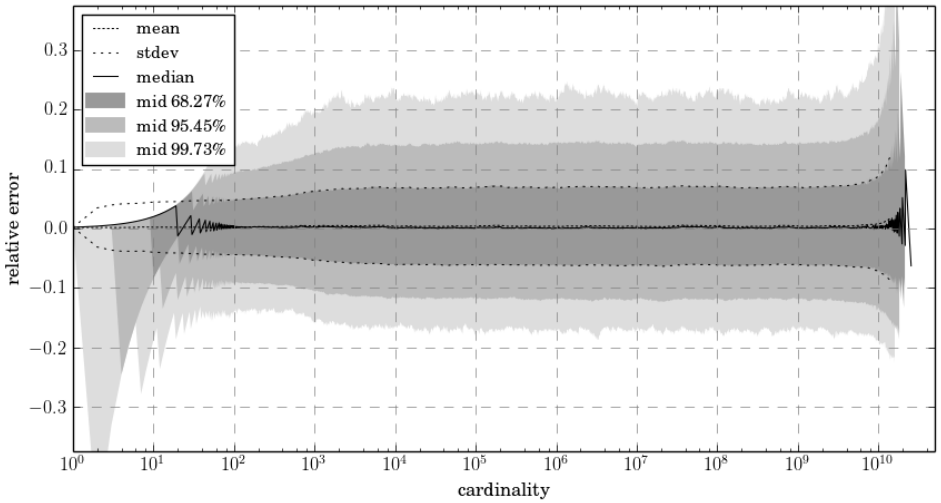

Figure 7: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 8$ and $q = 24$.

the ideal transition point. In contrast, the improved raw estimator performs well over the entire cardinality range.

The new estimation algorithm also works well for other HyperLogLog configurations. First we considered configurations using a 32-bit hash function ($p+q = 32$). The relative estimation error for precisions $p = 8$, $p = 16$, $p = 22$ are shown in Fig. 7, Fig. 8, and Fig. 9, respectively. As expected, since $p + q = 32$ is kept constant, the operating range remains more or less the same, while the relative error decreases with increasing precision. Again, the new algorithm gives essentially unbiased estimates. Only for very large precisions, an oscillating bias becomes apparent (compare Fig. 9), that is caused by approximating the periodic function ξ by a constant (see Section 3.1).

As proposed in [7], the operating range can be extended by replacing the 32-bit hash function by a 64-bit hash function. Fig. 10 shows the relative error for such a HyperLogLog configuration with parameters $p = 12$ and $q = 52$. The doubled hash value size shifts the maximum trackable cardinality value towards 2^{64}. As Fig. 10 shows, when compared to the 32-bit hash value case given in Fig. 5, the estimation error remains constant over the entire simulated cardinality range up to 50 billions.

We also evaluated the case $p = 12$ and $q = 14$, which is interesting, because the register values are limited to the range $[0, 15]$. As a consequence, 4 bits are sufficient for representing a single register value. This allows two registers to share a single byte, which is beneficial from a performance perspective. Nevertheless, this configuration still allows the estimation of cardinalities up to 100 millions as shown in Fig. 11, which could be enough for many applications.

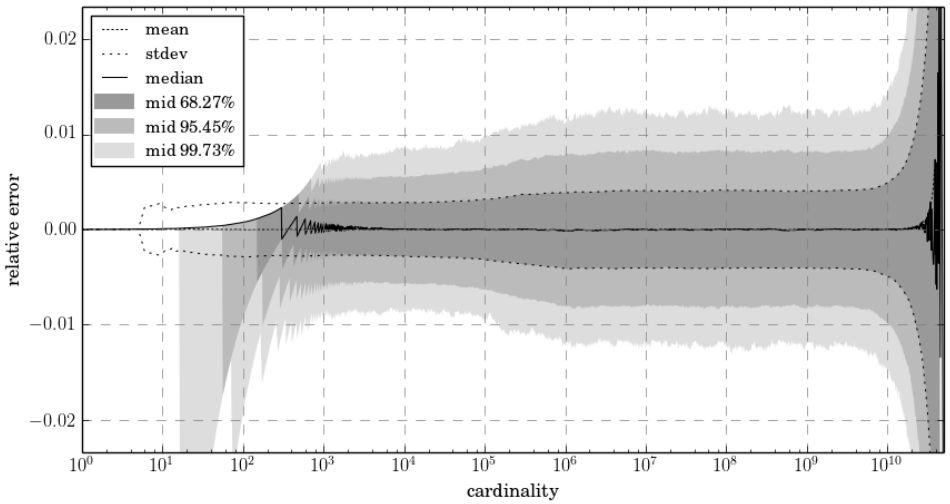

Figure 8: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 16$ and $q = 16$.

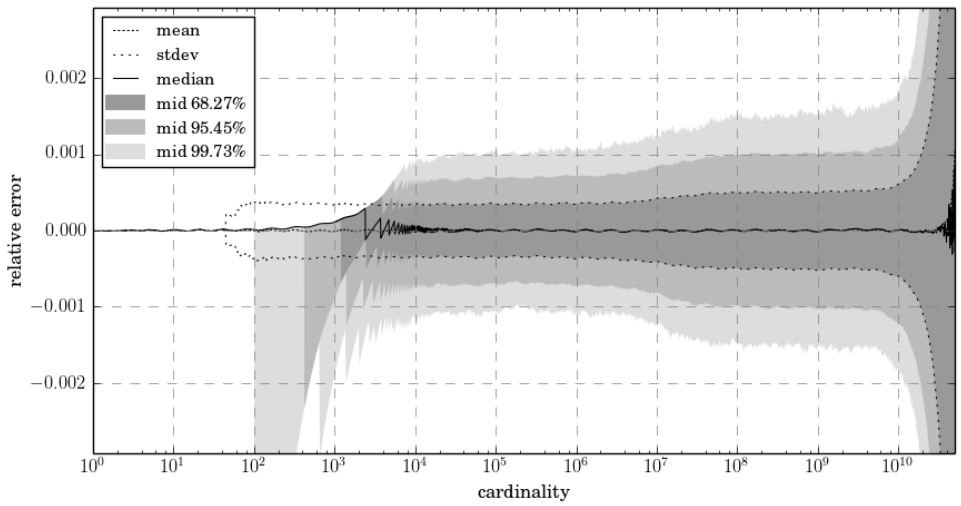

Figure 9: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 22$ and $q = 10$.

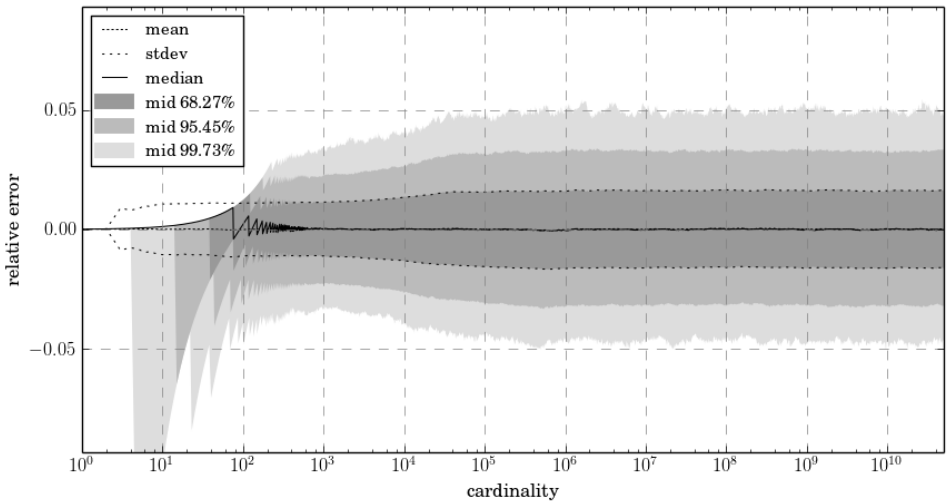

Figure 10: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 52$.

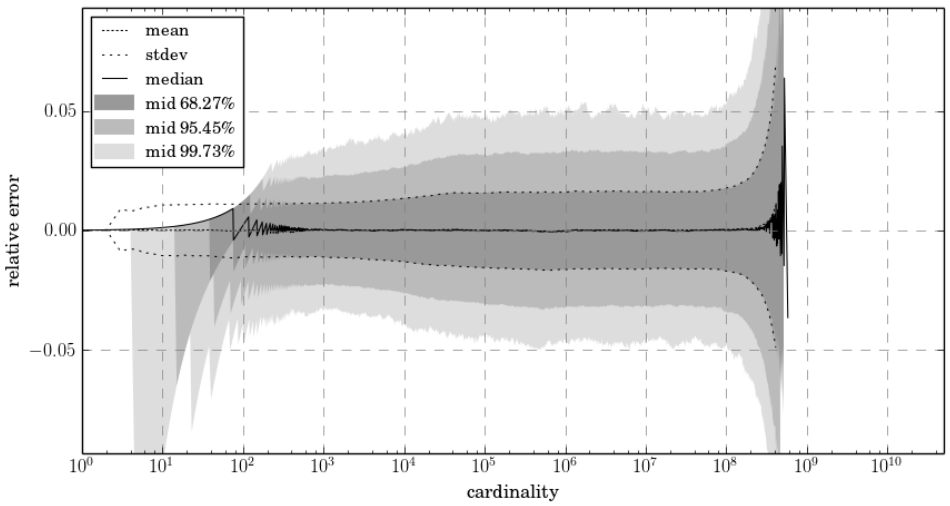

Figure 11: Relative error of the improved raw estimator as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 14$.

3.6. Performance

In order to evaluate the performance of the improved raw estimation algorithm, we investigated the average computation time for estimating the cardinality from a given HyperLogLog sketch. For different cardinalities we loaded precalculated multiplicity vectors of 1000 randomly generated HyperLogLog sketches into main memory. The average computation time was determined by cycling over these multiplicity vectors and passing them as input to the algorithm. For each evaluated cardinality value the average execution time was calculated after 100 cycles which corresponds to 100 000 algorithm executions for each cardinality value. The results for HyperLogLog configurations $p = 12, q = 20$ and $p = 12, q = 52$ are shown in Fig. 12. Two variants of Algorithm 4 have been evaluated for which the functions σ and τ have been either calculated on-demand or taken from a lookup table. All these benchmarks where carried out on an Intel Core i5-2500K clocking at 3.3 GHz.

The results show that the execution times are nearly constant for $q = 52$. Using a lookup table makes not much difference, because the on-demand calculation for σ is very fast and the calculation of τ is rarely needed due to the small probability of saturated registers, $P(C_{53} > 0) \approx 0$, for realistic cardinalities.

For the case $q = 20$ with precalculated correction values the computation time is again – as expected – independent of the cardinality. The faster computation times compared to the $q = 52$ case can be explained by the much smaller dimension of the multiplicity vector which is equal to $q + 2$. If the functions σ and τ are calculated on-demand, the execution times for larger cardinalities are doubled. This comes from the fact that the calculation of τ is much more expensive than that of σ, because it requires more iterations and involves square root evaluations. However, we can imagine that a better numerical approximation of τ can be found than that given in Algorithm 4 and which allows on-demand evaluation without significant extra costs.

The numbers do not yet include the required processing time to extract the multiplicity vector out from the HyperLogLog sketch, which requires a complete scan over all registers and counting the different register values into an array. A theoretical lower bound for this processing time can be derived using the maximum memory bandwidth of the CPU, which is 21 GB/s for an Intel Core i5-2500K. If we consider a HyperLogLog sketch with precision $p = 12$ which uses 5 bits per register, the total data size of the HyperLogLog sketch is 2.5 kB minimum. Consequently, the transfer time from main memory to CPU will be at least 120 ns. Having this value in mind, the presented numbers for estimating the cardinality from the multiplicity vector are quite satisfying.

4. Maximum likelihood estimation

We know from Section 2.4 that any unbiased estimator for the Poisson parameter is also an unbiased estimator for the cardinality. Moreover, we know that under suitable regularity conditions of the probability mass function the maximum likelihood estimator is asymptotically efficient [16]. This means, if the number of registers m is large enough, the maximum likelihood method should give us an unbiased estimator for the cardinality.

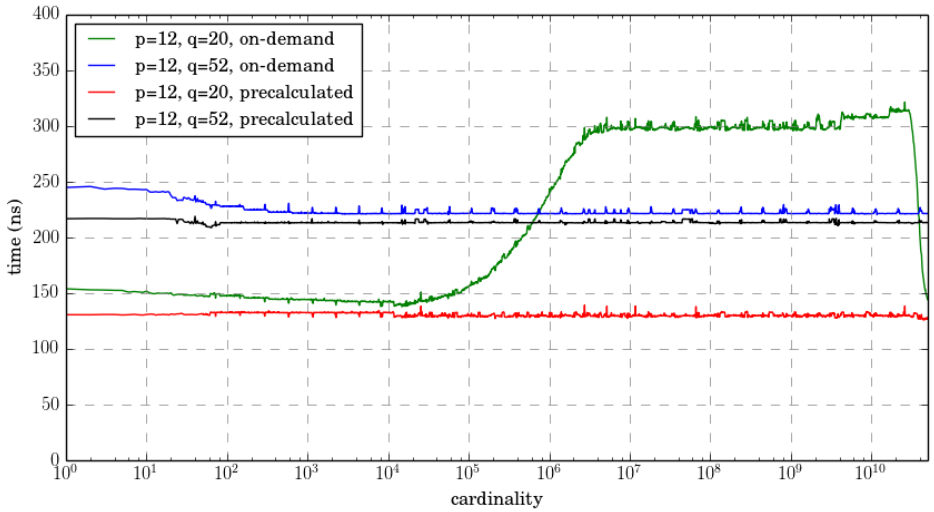

Figure 12: Average computation time as a function of the true cardinality with an Intel Core i5-2500K clocking at 3.3 GHz when estimating the cardinality from HyperLogLog sketches with parameters $p = 12$, $q = 20$ and $p = 12$, $q = 52$, respectively. Both cases, σ and τ precalculated and calculated on-demand have been considered.

For HyperLogLog sketches that have been obtained without stochastic averaging (see Section 2.1) the maximum likelihood method was already previously considered for cardinality estimation [17]. However, in this case the register values are statistically independent by nature which allows factorization of the joint probability mass function and a straightforward calculation of the maximum likelihood estimate. The practically more relevant stochastic averaging case, which is considered in this paper, gives a more complicated likelihood function. However, the Poisson approximation makes the maximum likelihood method feasible again and we are finally able to derive another new robust and efficient cardinality estimation algorithm. Furthermore, in the course of the derivation we will demonstrate that consequent application of the maximum likelihood method reveals that the cardinality estimate needs to be roughly proportional to the harmonic mean for intermediate cardinality values. The history of the HyperLogLog algorithm shows that the raw estimator (12) was first found after several attempts using the geometric mean [6, 15].

4.1. Log-likelihood function

Using the probability mass function of the Poisson model (6) the log-likelihood and its derivative are given by

$$\log \mathcal{L}(\lambda|\boldsymbol{K}) = -\frac{\lambda}{m}\sum_{k=0}^{q}\frac{C_k}{2^k} + \sum_{k=1}^{q} C_k \log\left(1 - e^{-\frac{\lambda}{m2^k}}\right) + C_{q+1}\log\left(1 - e^{-\frac{\lambda}{m2^q}}\right) \tag{37}$$

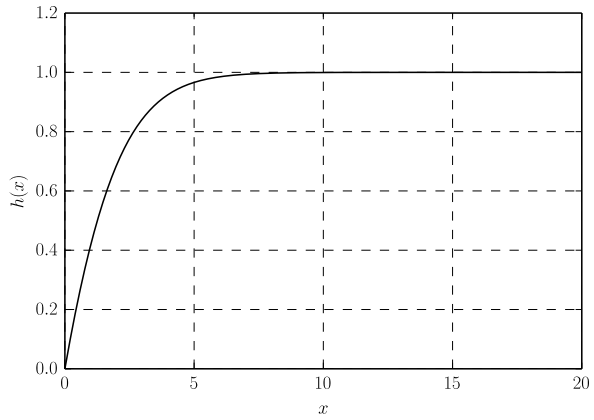

Figure 13: The function $h(x)$.

and

$$\frac{d}{d\lambda}\log\mathcal{L}(\lambda|\boldsymbol{K}) = -\frac{1}{\lambda}\left(\frac{\lambda}{m}\sum_{k=0}^{q}\frac{C_k}{2^k} + \sum_{k=1}^{q}C_k\frac{\frac{\lambda}{m2^k}}{1-e^{\frac{\lambda}{m2^k}}} + C_{q+1}\frac{\frac{\lambda}{m2^q}}{1-e^{\frac{\lambda}{m2^q}}}\right). \tag{38}$$

As a consequence, the maximum likelihood estimate for the Poisson parameter is given by

$$\hat{\lambda} = m\hat{x}, \tag{39}$$

if $\hat{x}$ denotes the root of the function

$$f(x) := x\sum_{k=0}^{q}\frac{C_k}{2^k} + \sum_{k=1}^{q}C_k\frac{\frac{x}{2^k}}{1-e^{\frac{x}{2^k}}} + C_{q+1}\frac{\frac{x}{2^q}}{1-e^{\frac{x}{2^q}}}. \tag{40}$$

This function can also be written as

$$f(x) := x\sum_{k=0}^{q}\frac{C_k}{2^k} + \sum_{k=1}^{q}C_k h\Big(\frac{x}{2^k}\Big) + C_{q+1}h\Big(\frac{x}{2^q}\Big) - (m - C_0)\,, \tag{41}$$

where the function $h(x)$ is defined as

$$h(x) := 1 - \frac{x}{e^x - 1}. \tag{42}$$

$h(x)$ is strictly increasing and concave as can be seen in Fig. 13. For non-negative values x the function ranges from $h(0) = 0$ to $h(x \to \infty) = 1$. Since the function $f(x)$ is also strictly increasing, it is obvious that there exists a unique root $\hat{x}$ for which $f(\hat{x}) = 0$. The function is non-positive at 0 since $f(0) = C_0 - m \leq 0$ and, in case $C_{q+1} < m$ which implies $\sum_{k=0}^{q}\frac{C_k}{2^k} > 0$, the function is at least linearly increasing. $C_{q+1} = m$ corresponds to the case with all registers equal to the maximum value $q+1$, for which the maximum likelihood estimate would be positive infinite.

It is easy to see that the estimate $\hat{\lambda}$ remains equal or becomes larger, when inserting an element into the HyperLogLog sketch following Algorithm 1. An update potentially changes the multiplicity vector $(C_0, \ldots, C_{q+1})$ to $(C_0, \ldots, C_i - 1, \ldots, C_j + 1, \ldots, C_{q+1})$ where $i < j$. Writing (41) as

$$f(x) := C_0 x + C_1 \left(h\Big(\frac{x}{2^1}\Big) + \frac{x}{2^1} - 1 \right) + C_2 \left(h\Big(\frac{x}{2^2}\Big) + \frac{x}{2^2} - 1 \right) + \ldots \\ \cdots + C_q \left(h\Big(\frac{x}{2^q}\Big) + \frac{x}{2^q} - 1 \right) + C_{q+1} \left(h\Big(\frac{x}{2^q}\Big) - 1 \right). \quad (43)$$

shows that the coefficient of C_i is larger than the coefficient of C_j in case $i < j$. Keeping x fixed during an update decreases $f(x)$. As a consequence, since $f(x)$ is increasing, the new root and hence the estimate must be larger than prior the update.

For the special case $q = 0$, which corresponds to the already mentioned linear counting algorithm, (40) can be solved analytically. In this case, the maximum likelihood method under the Poisson model leads directly to the linear counting estimator (15). Due to this fact we could expect that maximum likelihood estimation under the Poisson model also works well for the more general HyperLogLog case.

4.2. Inequalities for the maximum likelihood estimate

In the following lower and upper bounds for $\hat{x}$ are derived. Applying Jensen's inequality on h in (41) gives an upper bound for $f(x)$:

$$f(x) \leq x \sum_{k=0}^{q} \frac{C_k}{2^k} + (m - C_0) \cdot h\left(x \cdot \frac{\sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}}{m - C_0} \right) - (m - C_0). \quad (44)$$

The left-hand side is zero, if $\hat{x}$ is inserted. Resolution for $\hat{x}$ finally gives the lower bound

$$\hat{x} \geq \frac{m - C_0}{\sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}} \log\left(1 + \frac{\sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}}{\sum_{k=0}^{q} \frac{C_k}{2^k}} \right). \quad (45)$$

This bound can be weakened using $\log(1 + x) \geq \frac{2x}{x+2}$ for $x \geq 0$ which results in

$$\hat{x} \geq \frac{m - C_0}{C_0 + \frac{3}{2} \sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}}. \quad (46)$$

Using the monotonicity of h, the lower bound

$$f(x) \geq x \sum_{k=0}^{q} \frac{C_k}{2^k} + \sum_{k=1}^{q} C_k h\Big(\frac{x}{2^{K'_{\max}}}\Big) + C_{q+1} h\Big(\frac{x}{2^{K'_{\max}}}\Big) - (m - C_0) \quad (47)$$

can be found, where $K'_{\max} := \min(K_{\max}, q)$ and $K_{\max} := \max\{k | C_k > 0\}$. Again, inserting $\hat{x}$ and transformation gives

$$\hat{x} \leq 2^{K'_{\max}} \log\left(1 + \frac{m - C_0}{2^{K'_{\max}} \sum_{k=0}^{q} \frac{C_k}{2^k}} \right) \quad (48)$$

as upper bound which can be weakened using $\log(1+x) \leq x$ for $x \geq 0$

$$\hat{x} \leq \frac{m - C_0}{\sum_{k=0}^{q} \frac{C_k}{2^k}}. \tag{49}$$

If the HyperLogLog sketch is in the intermediate range, where $C_0 = C_{q+1} = 0$ the bounds (46) and (49) differ only by a constant factor from the raw estimator (12). Hence, consequent application of the maximum likelihood method leads directly to the harmonic mean that is used by the raw estimator.

4.3. Computation of the maximum likelihood estimate

Since f is concave and increasing, both, Newton-Raphson iteration and the secant method, will converge to the root, provided that the function is negative for the starting points. In the following we start from the secant method to derive the new cardinality estimation algorithm. Even though the secant method has the disadvantage of slower convergence, a single iteration is simpler to calculate as it does not require the evaluation of the first derivative. An iteration step of the secant method can be written as

$$x_i = x_{i-1} - (x_{i-1} - x_{i-2}) \frac{f(x_{i-1})}{f(x_{i-1}) - f(x_{i-2})}. \tag{50}$$

If we set x_0 equal to 0, for which $f(x_0) = -(m - C_0)$, and x_1 equal to one of the derived lower bounds (45) or (46), the sequence $(x_0, x_1, x_2, \ldots)$ is montone increasing. Using the definitions

$$\Delta x_i := x_i - x_{i-1} \tag{51}$$

and

$$g(x) := f(x) + (m - C_0) = x \sum_{k=0}^{q} \frac{C_k}{2^k} + \sum_{k=1}^{q} C_k h\Big(\frac{x}{2^k}\Big) + C_{q+1} h\Big(\frac{x}{2^q}\Big) \tag{52}$$

the iteration scheme can also be written as

$$\Delta x_i = \Delta x_{i-1} \frac{(m - C_0) - g(x_{i-1})}{g(x_{i-1}) - g(x_{i-2})}, \tag{53}$$

$$x_i = x_{i-1} + \Delta x_i. \tag{54}$$

The iteration can be stopped, if $\Delta x_i \leq \delta \cdot x_i$. Since the expected statistical errror for the HyperLogLog data structure scales according to $\frac{1}{\sqrt{m}}$ [6], it makes sense to choose $\delta = \frac{\varepsilon}{\sqrt{m}}$ with some constant ε. For all results presented later in Section 4.5 we have used $\varepsilon = 10^{-2}$.

4.4. Maximum likelihood estimation algorithm

In order to get a fast cardinality estimation algorithm, it is crucial to minimize evaluation costs for (52). A couple of optimizations allow significant reduction of computational effort:

- Only a fraction of all count values C_k is non-zero. If we denote $K_{\min} := \min\{k|C_k > 0\}$ and $K_{\max} := \max\{k|C_k > 0\}$, it is sufficient to loop over all indices in the range $[K_{\min}, K_{\max}]$.
- The sum $\sum_{k=0}^{q} \frac{C_k}{2^k}$ in (52) can be precalculated and reused for all function evaluations.
- Many programming languages allow the efficient multiplication and division by any integral power of two using special functions, such as `ldexp` in C/C++ or `scalb` in Java.
- The function $h(x)$ only needs to be evaluated at values $\left\{\frac{x}{2^{K'_{\max}}}, \frac{x}{2^{K'_{\max}-1}}, \ldots, \frac{x}{2^{K_{\min}}}\right\}$ where $K'_{\max} := \min(K_{\max}, q)$. This series corresponds to a geometric series with ratio two. A straightforward calculation using (42) is very expensive because of the exponential function. However, if we know $h\left(\frac{x}{2^{K'_{\max}}}\right)$ all other required function values can be easily obtained using the identity

$$h(4x) = \frac{x + h(2x)\,(1 - h(2x))}{x + (1 - h(2x))}. \tag{55}$$

  This recursive formula is stable in a sense that the relative error of $h(4x)$ is smaller than that of $h(2x)$ as shown in Appendix B.
- If x is smaller than 0.5, the function $h(x)$ can be well approximated by a Taylor series around $x = 0$

$$h(x) = \frac{x}{2} - \frac{x^2}{12} + \frac{x^4}{720} - \frac{x^6}{30240} + \mathcal{O}(x^8), \tag{56}$$

  which can be optimized for numerical evaluation using Estrin's scheme and $x' := \frac{x}{2}$ and $x'' := x'x'$

$$h(x) = x' - x''/3 + \left(x''x''\right)\left(1/45 - x''/472.5\right) + \mathcal{O}(x^8). \tag{57}$$

  The smallest argument for which h needs to be evaluated is $\frac{x}{2^{K'_{\max}}}$. If $C_{q+1} = 0$, we can find an upper bound for the smallest argument using (48)

$$\frac{x}{2^{K'_{\max}}} \leq \log\left(1 + \frac{\sum_{k=0}^{K'_{\max}} C_k}{2^{K'_{\max}} \sum_{k=0}^{K'_{\max}} \frac{C_k}{2^k}}\right) \leq \log 2 \approx 0.693. \tag{58}$$

  In practice, $\frac{x}{2^{K'_{\max}}} \leq 0.5$ is satisfied most of the time as long as only a few registers are saturated, that is $C_{q+1} \ll m$. In case $\frac{x}{2^{K'_{\max}}} > 0.5$, $h\left(\frac{x}{2^\kappa}\right)$ is calculated instead with $\kappa = 2 + \lfloor \log_2(x) \rfloor$. By definition, $\frac{x}{2^\kappa} \leq 0.5$ which allows using the Taylor series approximation. $h\left(\frac{x}{2^{K'_{\max}}}\right)$ is finally obtained after $\kappa - K'_{\max}$ iterations using (55). As shown in Appendix C, a small approximation error of h does not have much impact on the error of the maximum likelihood estimate as long as most registers are not yet saturated.

Putting all these optimizations together finally gives the new cardinality estimation algorithm presented as Algorithm 5. The algorithm requires mainly only elementary operations. The square root that appears for the calculation of the stopping limit $\delta = \varepsilon/\sqrt{m}$ can be precomputed, because it only depends on the HyperLogLog precision parameter p.

For very large cardinalities it makes sense to use the strong (45) instead of the weak lower bound (46) as second starting point for the secant method. The stronger bound is a much better approximation especially for large cardinalities, where the extra logarithm evaluation is amortized by savings in the number of iteration cycles. Therefore, the presented algorithm switches over to the stronger bound, if

$$\frac{\sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}}{\sum_{k=0}^{q} \frac{C_k}{2^k}} > 1.5 \tag{59}$$

is satisfied. The threshold value of 1.5 was found to be a reasonable choice in order to reduce the computation time for large cardinalities significantly.

4.5. Estimation error

In order to investigate the estimation error for the maximum likelihood estimation algorithm, we investigated the same HyperLogLog configurations as in Section 3.5 for the improved raw estimation algorithm. Fig. 14, Fig. 15, Fig. 16, Fig. 17, Fig. 18, and Fig. 19 show very similar results for various HyperLogLog parameters.

What is different for the maximum likelihood estimation approach is a somewhat smaller median bias with less oscillations around zero for small cardinalities. The standard deviation of the relative error is also slightly better for the maximum likelihood estimator than for the improved raw estimator as shown in Fig. 6. Furthermore, contrary to the raw estimator which reveals a small oscillating bias for the mean (see Fig. 9), the maximum likelihood estimator seems to be completely unbiased (see Fig. 17).

4.6. Performance

We also measured the performance of Algorithm 5 using the same test setup as described in Section 3.6. The results for HyperLogLog configurations $p = 12, q = 20$ and $p = 12, q = 52$ are shown in Fig. 20. The average computation time for the maximum likelihood algorithm shows a different behavior than for the improved raw estimation algorithm (compare Fig. 12).As can be seen, the average execution time is larger for most cardinalities, but nevertheless, it never exceeds 700 ns which is still fast enough for many applications. The steps in the chart can be explained by different numbers of iteration cycles until the secant method is stopped. For example, at a cardinality value around 5000 the average number of required cycles until the stop criterion is met increases abruptly from two to three. More than three iteration cycles have never been observed for any cardinality estimate in this performance test.

Algorithm 5 Maximum likelihood cardinality estimation.

function EstimateCardinality($\boldsymbol{C}$)
  $q \leftarrow \dim(\boldsymbol{C}) - 2$
  $K_{\min} \leftarrow \min\{k | C_k > 0\}$
  if $K_{\min} > q$ **then**
    return ∞
  end if
  $K'_{\min} \leftarrow \max(K_{\min}, 1)$
  $K_{\max} \leftarrow \max\{k | C_k > 0\}$
  $K'_{\max} \leftarrow \min(K_{\max}, q)$
  $z \leftarrow 0$
  $m' \leftarrow C_{q+1}$
  $y \leftarrow 2^{-K'_{\max}}$
  for $k \leftarrow K'_{\max}, K'_{\min}$ **do**
    $z \leftarrow z + C_k \cdot y$ ▷ here $y = 2^{-k}$
    $y \leftarrow 2y$
    $m' \leftarrow m' + C_k$
  end for
  ▷ here $z = \sum_{k=1}^{q} \frac{C_k}{2^k}$
  $m \leftarrow m' + C_0$
  $c \leftarrow C_{q+1}$
  if $q \geq 1$ **then**
    $c \leftarrow c + C_{K'_{\max}}$
  end if
  $g_{\text{prev}} \leftarrow 0$
  $a \leftarrow z + C_0$ ▷ $a = \sum_{k=0}^{q} \frac{C_k}{2^k}$
  $b \leftarrow z + C_{q+1} \cdot 2^{-q}$ ▷ $b = \sum_{k=1}^{q} \frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}$
  if $b \leq 1.5 \cdot a$ **then**
    $x \leftarrow m'/(0.5 \cdot b + a)$ ▷ weak lower bound (46)
  else
    $x \leftarrow m'/b \cdot \log(1 + b/a)$ ▷ strong lower bound (45)
  end if

Algorithm 5 Maximum likelihood cardinality estimation (continued).

$\Delta x \leftarrow x$
$\delta \leftarrow \varepsilon/\sqrt{m}$ ▷ $\varepsilon = 10^{-2}$ (see Section 4.3)
while $\Delta x > x \cdot \delta$ **do** ▷ secant method iteration
  $\kappa \leftarrow 2 + \lfloor \log_2(x) \rfloor$
  $x' \leftarrow x \cdot 2^{-\max(K'_{\max},\kappa)-1}$ ▷ $x' \in [0, 0.25]$
  $x'' \leftarrow x' \cdot x'$
  $h \leftarrow x' - x''/3 + (x'' \cdot x'') \cdot (1/45 - x''/472.5)$ ▷ Taylor approximation (57)
  for $k \leftarrow (\kappa - 1), K'_{\max}$ **do**
    $h \leftarrow \frac{x'+h\cdot(1-h)}{x'+(1-h)}$ ▷ calculate $h\left(\frac{x}{2^k}\right)$, see (55), at this point $x' = \frac{x}{2^{k+2}}$
    $x' \leftarrow 2x'$
  end for
  $g \leftarrow c \cdot h$ ▷ compare (52)
  for $k \leftarrow (K'_{\max} - 1), K'_{\min}$ **do**
    $h \leftarrow \frac{x'+h\cdot(1-h)}{x'+(1-h)}$ ▷ calculate $h\left(\frac{x}{2^k}\right)$, see (55), at this point $x' = \frac{x}{2^{k+2}}$
    $g \leftarrow g + C_k \cdot h$
    $x' \leftarrow 2x'$
  end for
  $g \leftarrow g + x \cdot a$
  if $g > g_{\text{prev}} \wedge m' \geq g$ **then**
    $\Delta x \leftarrow \Delta x \cdot \frac{m'-g}{g-g_{\text{prev}}}$ ▷ see (53)
  else
    $\Delta x \leftarrow 0$
  end if
  $x \leftarrow x + \Delta x$
  $g_{\text{prev}} \leftarrow g$
end while
return $m \cdot x$
end function

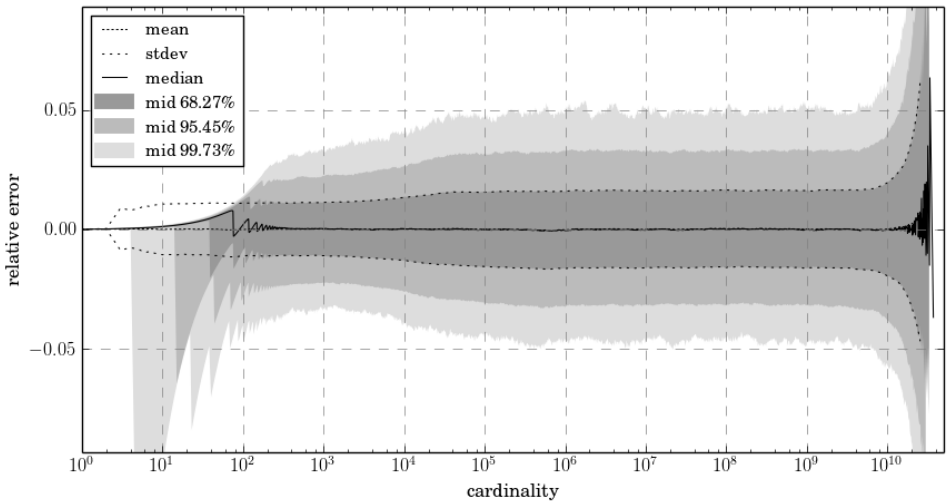

Figure 14: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 20$.

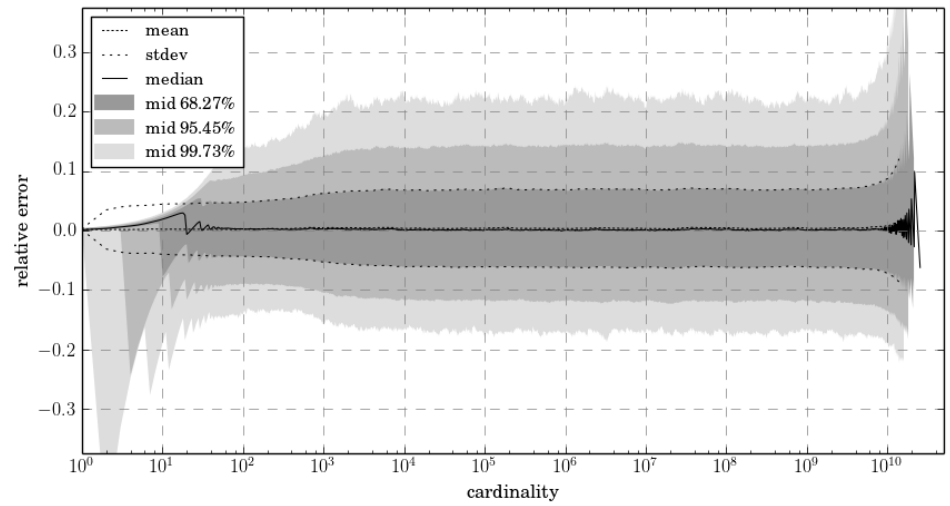

Figure 15: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 8$ and $q = 24$.

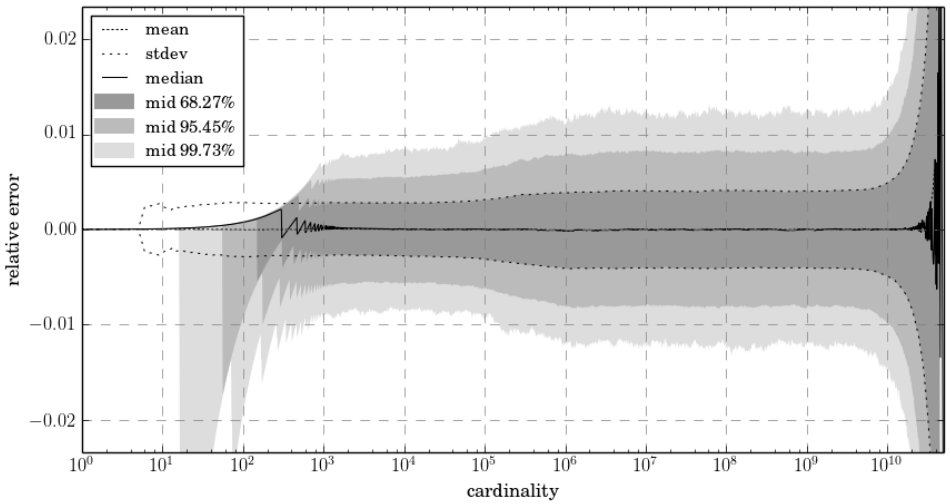

Figure 16: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 16$ and $q = 16$.

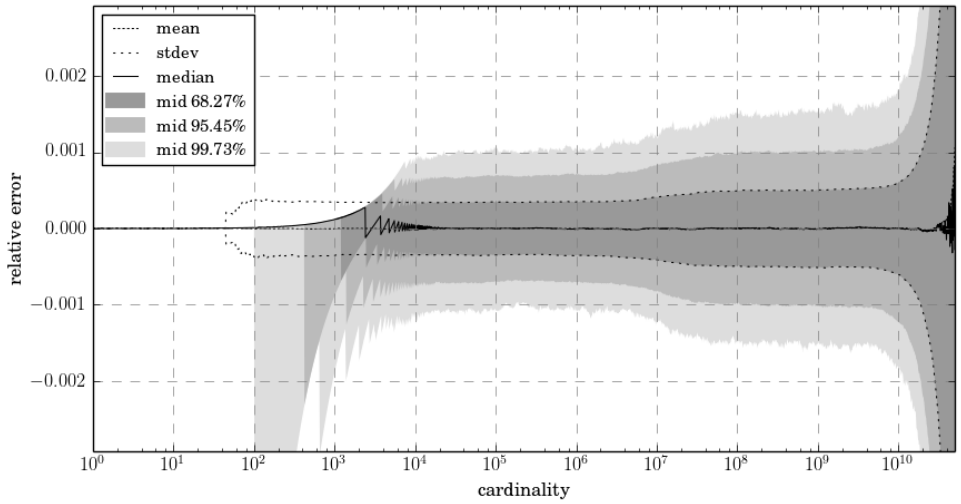

Figure 17: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 22$ and $q = 10$.

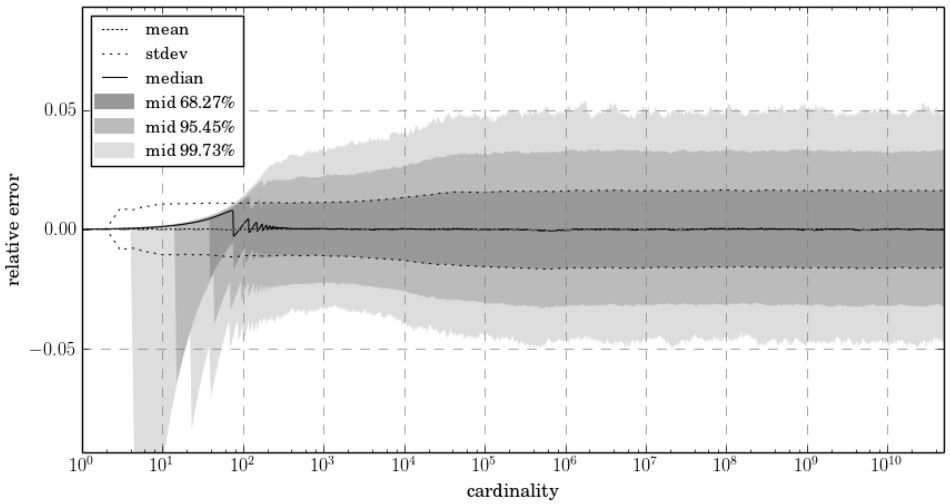

Figure 18: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 52$.

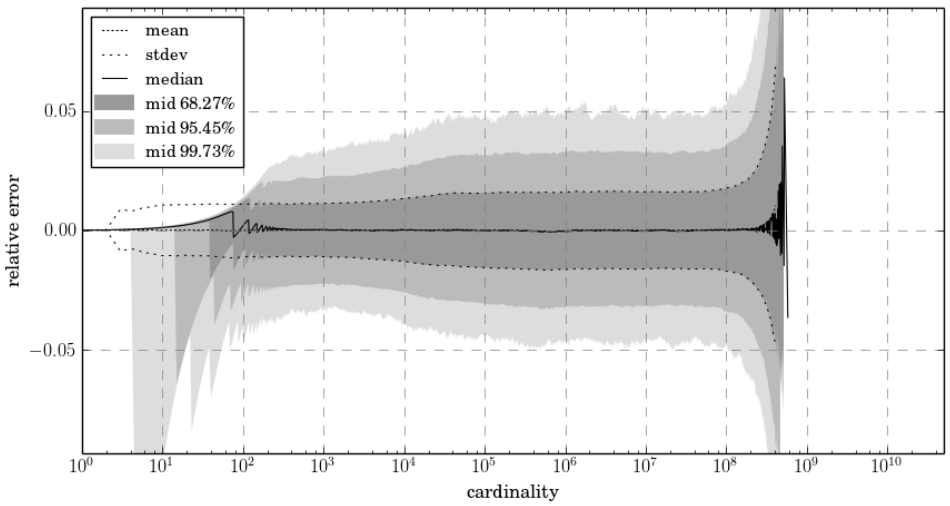

Figure 19: Relative error of the maximum likelihood estimates as a function of the true cardinality for a HyperLogLog sketch with parameters $p = 12$ and $q = 14$.

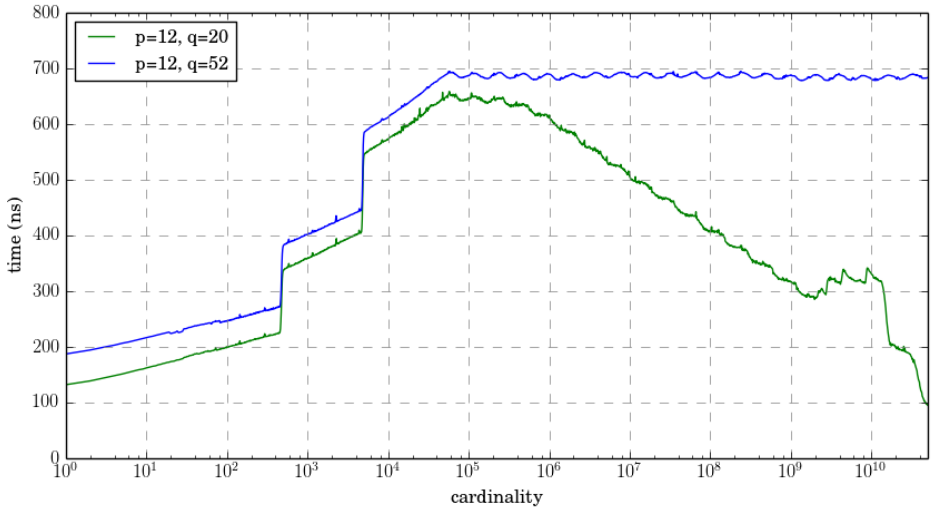

Figure 20: Average execution time of the maximum likelihood estimation algorithm as a function of the true cardinality with an Intel Core i5-2500K clocking at 3.3 GHz for HyperLogLog sketches with parameters $p = 12$, $q = 20$ and $p = 12$, $q = 52$, respectively.

5. Cardinality estimation of set intersections and complements

While the union of two sets that are represented by HyperLogLog sketches can be straightforwardly computed using (1), the computation of cardinalities of other set operations like intersections and complements is more challenging. The conventional approach uses the relationships

$$\begin{aligned} |S_1 \setminus S_2| &= |S_1 \cup S_2| - |S_2|\,, \\ |S_2 \setminus S_1| &= |S_1 \cup S_2| - |S_1|\,, \\ |S_1 \cap S_2| &= |S_1| + |S_2| - |S_1 \cup S_2|\,, \end{aligned} \tag{60}$$

and the fact that HyperLogLog sketches can be easily merged using (1). The last equation is also known as inclusion-exclusion principle and can be used to calculate the intersection size. Unfortunately, the estimation error does not scale well for the inclusion-exclusion principle. Especially for small Jaccard indices, the relative estimation error can become very large [18]. In the worst case, the estimate could be negative without artificial restriction to nonnegative values. Therefore, it was proposed to combine HyperLogLog sketches with minwise hashing [19, 20], which improves the estimation error, even though at the expense of significant more space consumption.

It was recently pointed out without special focus on HyperLogLog sketches, that the application of the maximum likelihood method to the joint likelihood function of two probabilistic data structures, which represent the intersection operands, gives better intersection size estimates [21]. For HyperLogLog sketches recorded without stochastic averaging (compare Section 2.1) this was shown in [20], where also an algorithm based

on the maximum likelihood principle was outlined. However, in practice, HyperLogLog sketches are recorded using stochastic averaging because of the much cheaper element insertions as described by Algorithm 1. Motivated by the good results we have obtained for a single HyperLogLog sketch using the maximum likelihood method in combination with the Poisson approximation, we are tempted to apply this approach also for the estimation of set operation result sizes.

Assume two given HyperLogLog sketches with register values $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ representing the sets S_1 and S_2, respectively. The goal is to find estimates for the cardinalities of the pairwise disjoint sets $X = S_1 \cap S_2$, $A = S_1 \setminus S_2$, and $B = S_2 \setminus S_1$. The Poisson approximation allows us to assume that pairwise distinct elements are inserted into the HyperLogLog sketches representing S_1 and S_2 at rates λ_a and λ_b, respectively. Furthermore, we assume that further unique elements are inserted into both HyperLogLog sketches simultaneously at rate λ_x. We expect that good estimates $\hat{\lambda}_a$, $\hat{\lambda}_b$, and $\hat{\lambda}_x$ for the rates are also good estimates for the cardinalities of A, B, and X.

5.1. Joint log-likelihood funtion

In order to get maximum likelihood estimators for $\hat{\lambda}_a$, $\hat{\lambda}_b$, and $\hat{\lambda}_x$ we need to derive the joint probability distribution of two HyperLogLog sketches. Under the Poisson model the register values are independent and identically distributed. Therefore, we first derive the joint probability distribution for a single register that has value K_1 in the first HyperLogLog sketch representing S_1 and value K_2 in the second HyperLogLog sketch representing S_2.

The HyperLogLog sketch that represents S_1 can be thought to be constructed from two HyperLogLog sketches representing A and X and merging both using (1). Analogously, the HyperLogLog sketch for S_2 could have been obtained from sketches for B and X. Let K_a, K_b, and K_x be the value of the considered register in the HyperLogLog sketch representing A, B, and X, respectively. The corresponding values in sketches for S_1 and S_2 are given by

$$K_1 = \max(K_a, K_x)\,, \quad K_2 = \max(K_b, K_x)\,. \tag{61}$$

Their joint cumulative probability function is given as

$$\begin{aligned} P(K_1 \le k_1 \wedge K_2 \le k_2) &= P(\max(K_a, K_x) \le k_1 \wedge \max(K_b, K_x) \le k_2) \\ &= P(K_a \le k_1 \wedge K_b \le k_2 \wedge K_x \le \min(k_1, k_2)) \\ &= P(K_a \le k_1)\, P(K_b \le k_2)\, P(K_x \le \min(k_1, k_2))\,. \end{aligned} \tag{62}$$

Here the last transformation used the independence of K_a, K_b, and K_x, because by definition, the sets A, B, and X are pairwise disjoint. Furthermore, under the Poisson model K_a, K_b, and K_x obey (7). If we take into account that pairwise distinct elements are added to A, B, and X at rates λ_x, λ_a, and λ_b, respectively, the probability that a certain register has a value less than or equal to k_1 in the first HyperLogLog sketch and

simultaneously a value less than or equal to k_2 in the second one can be written as

$$P(K_1 \leq k_1 \wedge K_2 \leq k_2) = \begin{cases} 0 & k_1 < 0 \vee k_2 < 0 \\ e^{-\frac{\lambda_a}{m2^{k_1}} - \frac{\lambda_b}{m2^{k_2}} - \frac{\lambda_x}{m2^{\min(k_1,k_2)}}} & 0 \leq k_1 \leq q \wedge 0 \leq k_2 \leq q \\ e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}}} & 0 \leq k_2 \leq q < k_1 \\ e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}}} & 0 \leq k_1 \leq q < k_2 \\ 1 & q < k_1 \wedge q < k_2. \end{cases} \tag{63}$$

The joint probability mass function for both register values can be calculated using

$$\begin{aligned} \rho(k_1, k_2) = P(K_1 \leq k_1 \wedge K_2 \leq k_2) - P(K_1 \leq k_1 - 1 \wedge K_2 \leq k_2) \\ - P(K_1 \leq k_1 \wedge K_2 \leq k_2 - 1) + P(K_1 \leq k_1 - 1 \wedge K_2 \leq k_2 - 1), \end{aligned} \tag{64}$$

which finally gives

$$\rho(k_1, k_2) = \begin{cases} e^{-\frac{\lambda_a+\lambda_x}{m} - \frac{\lambda_b}{m2^{k_2}}} \left(1 - e^{-\frac{\lambda_b}{m2^{k_2}}}\right) & 0 = k_1 < k_2 \leq q \\ e^{-\frac{\lambda_a+\lambda_x}{m}} \left(1 - e^{-\frac{\lambda_b}{m2^q}}\right) & 0 = k_1 < k_2 = q + 1 \\ e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}} - \frac{\lambda_b}{m2^{k_2}}} \left(1 - e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}}}\right) \left(1 - e^{-\frac{\lambda_b}{m2^{k_2}}}\right) & 1 \leq k_1 < k_2 \leq q \\ e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}}} \left(1 - e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}}}\right) \left(1 - e^{-\frac{\lambda_b}{m2^q}}\right) & 1 \leq k_1 < k_2 = q + 1 \\ e^{-\frac{\lambda_b+\lambda_x}{m} - \frac{\lambda_a}{m2^{k_1}}} \left(1 - e^{-\frac{\lambda_a}{m2^{k_1}}}\right) & 0 = k_2 < k_1 \leq q \\ e^{-\frac{\lambda_b+\lambda_x}{m}} \left(1 - e^{-\frac{\lambda_a}{m2^q}}\right) & 0 = k_2 < k_1 = q + 1 \\ e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}} - \frac{\lambda_a}{m2^{k_1}}} \left(1 - e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}}}\right) \left(1 - e^{-\frac{\lambda_a}{m2^{k_1}}}\right) & 1 \leq k_2 < k_1 \leq q \\ e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}}} \left(1 - e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}}}\right) \left(1 - e^{-\frac{\lambda_a}{m2^q}}\right) & 1 \leq k_2 < k_1 = q + 1 \\ e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m}} & 0 = k_1 = k_2 \\ e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^k}} \left(1 - e^{-\frac{\lambda_a+\lambda_x}{m2^k}} - e^{-\frac{\lambda_b+\lambda_x}{m2^k}} + e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^k}}\right) & 1 \leq k_1 = k_2 = k \leq q \\ 1 - e^{-\frac{\lambda_a+\lambda_x}{m2^q}} - e^{-\frac{\lambda_b+\lambda_x}{m2^q}} + e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^q}} & k_1 = k_2 = q + 1. \end{cases} \tag{65}$$

The logarithm of the joint probability mass function can be written using Iverson

bracket notation ([true] := 1, [false] := 0) as

$$\begin{aligned}
&\log(\rho(k_1,k_2)) = \\
&\qquad \log\left(1-e^{-\frac{\lambda_a+\lambda_x}{m2^{k_1}}}\right)[1\le k_1<k_2] + \log\left(1-e^{-\frac{\lambda_a}{m2^{\min(k_1,q)}}}\right)[k_2<k_1] \\
&\quad + \log\left(1-e^{-\frac{\lambda_b+\lambda_x}{m2^{k_2}}}\right)[1\le k_2<k_1] + \log\left(1-e^{-\frac{\lambda_b}{m2^{\min(k_2,q)}}}\right)[k_1<k_2] \\
&\quad + \log\left(1-e^{-\frac{\lambda_a+\lambda_x}{m2^{\min(k_1,q)}}}-e^{-\frac{\lambda_b+\lambda_x}{m2^{\min(k_1,q)}}}+e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^{\min(k_1,q)}}}\right)[1\le k_1=k_2] \\
&\quad -\frac{\lambda_a}{m2^{k_1}}[k_1\le q] - \frac{\lambda_b}{m2^{k_2}}[k_2\le q] - \frac{\lambda_x}{m2^{\min(k_1,k_2)}}[k_1\le q\vee k_2\le q].
\end{aligned} \tag{66}$$

Since the values for different registers are independent under the Poisson model, we are now able to write the joint probability mass function for all registers in both HyperLogLog sketches

$$\rho(\boldsymbol{k}_1,\boldsymbol{k}_2) = \prod_{i=1}^{m}\rho(k_{1i},k_{2i}). \tag{67}$$

In order to get the maximum likelihood estimates $\hat{\lambda}_a$, $\hat{\lambda}_b$, and $\hat{\lambda}_x$ we need to maximize the log-likelihood function given by

$$\log\mathcal{L}(\lambda_a,\lambda_b,\lambda_x|\boldsymbol{K}_1,\boldsymbol{K}_2) = \sum_{i=1}^{m}\log(\rho(K_{1i},K_{2i})). \tag{68}$$

Insertion of (66) results in

$$\begin{aligned}
&\log\mathcal{L}(\lambda_a,\lambda_b,\lambda_x|\boldsymbol{K}_1,\boldsymbol{K}_2) = \\
&\quad \sum_{k=1}^{q}\log\left(1-e^{-\frac{\lambda_a+\lambda_x}{m2^k}}\right)C_{1k}^{<} + \log\left(1-e^{-\frac{\lambda_b+\lambda_x}{m2^k}}\right)C_{2k}^{<} \\
&+\sum_{k=1}^{q+1}\log\left(1-e^{-\frac{\lambda_a}{m2^{\min(k,q)}}}\right)C_{1k}^{>} + \log\left(1-e^{-\frac{\lambda_b}{m2^{\min(k,q)}}}\right)C_{2k}^{>} \\
&+\sum_{k=1}^{q+1}\log\left(1-e^{-\frac{\lambda_a+\lambda_x}{m2^{\min(k,q)}}}-e^{-\frac{\lambda_b+\lambda_x}{m2^{\min(k,q)}}}+e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^{\min(k,q)}}}\right)C_{k}^{=} \\
&-\frac{\lambda_a}{m}\sum_{k=0}^{q}\frac{C_{1k}^{<}+C_{k}^{=}+C_{1k}^{>}}{2^k} - \frac{\lambda_b}{m}\sum_{k=0}^{q}\frac{C_{2k}^{<}+C_{k}^{=}+C_{2k}^{>}}{2^k} - \frac{\lambda_x}{m}\sum_{k=0}^{q}\frac{C_{1k}^{<}+C_{k}^{=}+C_{2k}^{<}}{2^k},
\end{aligned} \tag{69}$$

where the constants $C_{1k}^{<}$, $C_{1k}^{>}$, $C_{2k}^{<}$, $C_{2k}^{>}$, and $C_{k}^{=}$ are defined as

$$\begin{aligned}
C_{1k}^{<} &:= |\{i|k=K_{1i}<K_{2i}\}|, \\
C_{1k}^{>} &:= |\{i|k=K_{1i}>K_{2i}\}|, \\
C_{2k}^{<} &:= |\{i|k=K_{2i}<K_{1i}\}|, \\
C_{2k}^{>} &:= |\{i|k=K_{2i}>K_{1i}\}|, \\
C_{k}^{=} &:= |\{i|k=K_{1i}=K_{2i}\}|.
\end{aligned} \tag{70}$$

These $5m$ values are sufficient for estimating λ_a, λ_b, and λ_x. Actually, the set of these constants can be further reduced, because $C_{10}^{>} = C_{20}^{>} = C_{1,q+1}^{<} = C_{2,q+1}^{<} = 0$ always holds.

Since (69) is the generalization of (37) for two HyperLogLog sketches, the log-likelihood function for a single HyperLogLog sketch can be obtained by considering special cases. For example, assuming $\lambda_x = 0$, which means that both sketches represent disjoint sets, (69) can be splitted into the sum of two unary functions with parameters λ_a and λ_b each of which correspond to (37) as expected. Or, consider two sketches representing identical sets. In this case all registers are equal, which means that $C_{1k}^{>} = C_{2k}^{>} = C_{1k}^{<} = C_{2k}^{<} = 0$ for all k, and therefore the maximum likelihood method yields $\hat{\lambda}_a = \hat{\lambda}_b = 0$ and the value for $\hat{\lambda}_x$ will be equal to the single HyperLogLog maximum likelihood estimate (37).

The log-likelihood function (69) does not always have a strict global maximum point. For example, if all register values of the first HyperLogLog sketch are larger than the corresponding values in the second HyperLogLog sketch, that is $C_{1k}^{<} = C_{k}^{=} = C_{2k}^{>} = 0$ for all k, the function can be rewritten as sum of two functions, one dependent on λ_a and the other dependent on $\lambda_b + \lambda_x$. The maximum is obtained, if $\lambda_a = \hat{\lambda}_1$ and $\lambda_b + \lambda_x = \hat{\lambda}_2$. Here $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are the maximum likelihood cardinality estimates for the first and second HyperLogLog sketch, respectively. This means that the maximum likelihood method makes no clear statement about the intersection size in this case. The estimate for λ_x could be anything between 0 and $\hat{\lambda}_2$. For comparison, the inclusion-exclusion approach would give $\hat{\lambda}_2$ as estimate. This result is questionable, because there is no evidence in this case, that the sets S_1 and S_2 really have common elements.

The inclusion-exclusion method does not use all the available information given by the sufficient statistic (70), because the estimator is a function of the three vectors $(\boldsymbol{C}_1^{<} + \boldsymbol{C}^{=} + \boldsymbol{C}_1^{>})$, $(\boldsymbol{C}_2^{<} + \boldsymbol{C}^{=} + \boldsymbol{C}_2^{>})$, and $(\boldsymbol{C}_1^{>} + \boldsymbol{C}^{=} + \boldsymbol{C}_2^{>})$. In contrast, the maximum likelihood method incorporates each value of the sufficient statistic (70) individually.

5.2. Computation of the maximum likelihood estimates

The maximum likelihood estimates can be obtained by maximizing (69). Since the three parameters are all non-negative, this is a constrained optimization problem. In order to get rid of these constraints, we use the transformation $\lambda = me^{\varphi}$. This mapping has also the nice property that relative accuracy limits are translated into absolute ones, because $\Delta\varphi = \Delta\lambda/\lambda$. Many optimization algorithm implementations allow the definition of absolute limits rather than relative ones.

The transformed log-likelihood function can be written as

$$
\begin{aligned}
&f(\varphi_a, \varphi_b, \varphi_x) := \log \mathcal{L}(me^{\varphi_a}, me^{\varphi_b}, me^{\varphi_x} | \boldsymbol{K}_1, \boldsymbol{K}_2) = \\
&+ \sum_{k=1}^{q} C_{1k}^{<} \log(z_{xk} + y_{xk} z_{ak}) + C_{2k}^{<} \log(z_{xk} + y_{xk} z_{bk}) \\
&+ \sum_{k=1}^{q} C_{1k}^{>} \log(z_{ak}) + C_{2k}^{>} \log(z_{bk}) + C_{k}^{=} \log(z_{xk} + y_{xk} z_{ak} z_{bk}) \\
&+ C_{1,q+1}^{>} \log(z_{aq}) + C_{2,q+1}^{>} \log(z_{bq}) + C_{q+1}^{=} \log(z_{xq} + y_{xq} z_{aq} z_{bq}) \\
&- \sum_{k=0}^{q} \left(C_{1k}^{<} + C_{k}^{=} + C_{1k}^{>}\right) x_{ak} + \left(C_{2k}^{<} + C_{k}^{=} + C_{2k}^{>}\right) x_{bk} + \left(C_{1k}^{<} + C_{k}^{=} + C_{2k}^{<}\right) x_{xk}.
\end{aligned}
\tag{71}
$$

Here we introduced the following expressions for simplification:

$$
x_{*k} := \frac{e^{\varphi_*}}{2^k}, \quad y_{*k} := e^{-x_{*k}}, \quad z_{*k} := 1 - y_{*k}. \tag{72}
$$

Quasi-Newton methods are commonly used for finding the maximum of such multi-dimensional functions. They require the calculation of the gradient $\nabla f = \left(\frac{\partial f}{\partial \varphi_a}, \frac{\partial f}{\partial \varphi_b}, \frac{\partial f}{\partial \varphi_x}\right)$ which is, using

$$
\frac{\partial x_{*k}}{\partial \varphi_*} = x_{*k}, \quad \frac{\partial y_{*k}}{\partial \varphi_*} = -x_{*k} y_{*k}, \quad \frac{\partial z_{*k}}{\partial \varphi_*} = x_{*k} y_{*k}, \tag{73}
$$

given by

$$
\begin{aligned}
\frac{\partial f}{\partial \varphi_a} =& \sum_{k=1}^{q} C_{1k}^{<} \frac{y_{xk} x_{ak} y_{ak}}{z_{xk} + y_{xk} z_{ak}} + C_{k}^{=} \frac{y_{xk} x_{ak} y_{ak} z_{bk}}{z_{xk} + y_{xk} z_{ak} z_{bk}} + C_{1k}^{>} \frac{x_{ak} y_{ak}}{z_{ak}} \\
&+ C_{q+1}^{=} \frac{y_{xq} x_{aq} y_{aq} z_{bq}}{z_{xq} + y_{xq} z_{aq} z_{bq}} + C_{1,q+1}^{>} \frac{x_{aq} y_{aq}}{z_{aq}} - \sum_{k=0}^{q} \left(C_{1k}^{<} + C_{k}^{=} + C_{1k}^{>}\right) x_{ak}, \\
\frac{\partial f}{\partial \varphi_b} =& \sum_{k=1}^{q} C_{2k}^{<} \frac{y_{xk} x_{bk} y_{bk}}{z_{xk} + y_{xk} z_{bk}} + C_{k}^{=} \frac{y_{xk} z_{ak} x_{bk} y_{bk}}{z_{xk} + y_{xk} z_{ak} z_{bk}} + C_{2k}^{>} \frac{x_{bk} y_{bk}}{z_{bk}} \\
&+ C_{q+1}^{=} \frac{y_{xq} z_{aq} x_{bq} y_{bq}}{z_{xq} + y_{xq} z_{aq} z_{bq}} + C_{2,q+1}^{>} \frac{x_{bq} y_{bq}}{z_{bq}} - \sum_{k=0}^{q} \left(C_{2k}^{<} + C_{k}^{=} + C_{2k}^{>}\right) x_{bk}, \\
\frac{\partial f}{\partial \varphi_x} =& \sum_{k=1}^{q} C_{1k}^{<} \frac{x_{xk} y_{xk} y_{ak}}{z_{xk} + y_{xk} z_{ak}} + C_{k}^{=} \frac{x_{xk} y_{xk} \left(y_{ak} + z_{ak} y_{bk}\right)}{z_{xk} + y_{xk} z_{bk} z_{ak}} + C_{2k}^{<} \frac{x_{xk} y_{xk} y_{bk}}{z_{xk} + y_{xk} z_{bk}} \\
&+ C_{q+1}^{=} \frac{x_{xq} y_{xq} \left(y_{aq} + z_{aq} y_{bq}\right)}{z_{xq} + y_{xq} z_{aq} z_{bq}} - \sum_{k=0}^{q} \left(C_{1k}^{<} + C_{k}^{=} + C_{2k}^{<}\right) x_{xk}.
\end{aligned}
\tag{74}
$$

The calculation of (71) and its derivatives requires some care when calculating y_{*k} and z_{*k}. Since x_{*k} is nonnegative, we have $y_{*k}, z_{*k} \in [0, 1]$. In order to reduce the numerical

error of z_{*k} for small x_{*k}, it is essential to use the function $\texttt{expm1}(x) := e^x - 1$ that is available in most programming languages. If x_{*k} is smaller than $\log(2)$, we calculate y_{*k} and z_{*k} via $z_{*k} = -\texttt{expm1}(-x_{*k})$ and $y_{*k} = 1 - z_{*k}$ and not as defined in (72). In this way the numerical error of both, y_{*k} and z_{*k}, is minimized and still only a single exponential function needs to be evaluated.

Apart from that, the numerical evaluation of (71) is straightforward. The arguments of all logarithms are in range $[0, 1]$. The case that some argument vanishes, which would cause the logarithm to be negative infinite, does not occur in practice. Consider for example the logarithm evaluation associated with $C_{1k}^{<}$ which is only relevant if $C_{1k}^{<} > 0$. In this case however, we can be certain that the cardinality of $A \cup X$ is at least 1. Therefore, it is expected that at least one of the two maximum likelihood estimates $\hat{\lambda}_a$ and $\hat{\lambda}_x$ is at least in the order of 1. If the optimization algorithm starts with appropriate initial values, function evaluations for which $\max(\lambda_a, \lambda_x) \ll 1$ are not expected. Furthermore, the argument of the logarithm can be bounded by $z_{xk} + y_{xk} z_{ak} \geq \max(z_{xk}, z_{ak}) = 1 - \exp\left(-\frac{\max(\lambda_a, \lambda_x)}{m 2^k}\right) \geq \min\left(\frac{1}{2}, \frac{\max(\lambda_a, \lambda_x)}{m 2^{k+1}}\right)$. For the last inequality we used $1 - e^{-x} \geq \frac{1}{2}\min(1, x)$. The derived lower bound shows that the argument of the logarithm is large enough to be accurately represented by double-precision floating-point numbers, provided that λ_a and λ_x are not both substantially smaller than 1. Similar argumentation holds for all other logarithmic terms and also the divisions that appear in (74).

Algorithm 6 demonstrates the calculation of the estimates given $\boldsymbol{C}_1^{>}$, $\boldsymbol{C}_1^{<}$, $\boldsymbol{C}_2^{>}$, $\boldsymbol{C}_2^{<}$, and $\boldsymbol{C}^{=}$. First, a case is distinguished, where all registers have a value equal to zero in at least one of both HyperLogLog sketches, that is $C_{10}^{<} + C_{0}^{=} + C_{20}^{<} = m$. In this case it is certain that the HyperLogLog sketches represent disjoint sets and their corresponding cardinality estimates can be used for $\hat{\lambda}_a$ and $\hat{\lambda}_b$, respectively.

For the general case the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [22] can be applied, which is a very popular algorithm for non-linear optimization problems. In particular, we used the implementation provided by the Dlib C++ library [23].

5.3. Results

In order to investigate the estimation error of the joint cardinality estimation algorithm Algorithm 6 we constructed HyperLogLog sketch pairs with parameters $p = 20$ and $q = 44$ and with known cardinalities $|A|$, $|B|$, and $|X|$. This is done by adding $|A|$ and $|B|$ random values to the first and second HyperLogLog sketch, respectively. Afterwards $|X|$ random values are added to both sketches. Table 1 lists all the different cases we have evaluated. For each case 3333 different HyperLogLog sketch pairs were constructed and passed to the joint cardinality estimation algorithm. Table 1 also shows the average number of iterations of the BFGS algorithm until the stop criterion was satisfied. Each iteration step involved a function (71) and a gradient (74) evaluation.

Table 2, Table 3, and Table 4 compare the relative estimation error to the conventional approach using single sketch cardinality estimation together with (60). The mean, the standard deviation, and the root-mean-square error are given. Furthermore, we calcu-

Algorithm 6 Joint cardinality estimation.

function EstimateCardinalities($\boldsymbol{C}_1^{>}$, $\boldsymbol{C}_1^{<}$, $\boldsymbol{C}_2^{>}$, $\boldsymbol{C}_2^{<}$, $\boldsymbol{C}^{=}$)
  $m \leftarrow \left\| \boldsymbol{C}_1^{<} + \boldsymbol{C}^{=} + \boldsymbol{C}_1^{>} \right\|_1$
  $\hat{\lambda}_{ax} \leftarrow$ EstimateCardinality$(\boldsymbol{C}_1^{<} + \boldsymbol{C}^{=} + \boldsymbol{C}_1^{>})$
  $\hat{\lambda}_{bx} \leftarrow$ EstimateCardinality$(\boldsymbol{C}_2^{<} + \boldsymbol{C}^{=} + \boldsymbol{C}_2^{>})$
  if $C_{10}^{<} + C_{0}^{=} + C_{20}^{<} = m$ **then**
    $\hat{\lambda}_a \leftarrow \hat{\lambda}_{ax},\ \hat{\lambda}_b \leftarrow \hat{\lambda}_{bx},\ \hat{\lambda}_x \leftarrow 0$ ▷ in this case $S_1 \cap S_2 = X = \emptyset$ holds
    return $(\hat{\lambda}_a, \hat{\lambda}_b, \hat{\lambda}_x)$
  end if
  $\hat{\lambda}_{abx} \leftarrow$ EstimateCardinality$(\boldsymbol{C}_1^{>} + \boldsymbol{C}^{=} + \boldsymbol{C}_2^{>})$
  $\varphi_a \leftarrow \log(\max(1, \hat{\lambda}_{abx} - \hat{\lambda}_{bx})/m)$
  $\varphi_b \leftarrow \log(\max(1, \hat{\lambda}_{abx} - \hat{\lambda}_{ax})/m)$
  $\varphi_x \leftarrow \log(\max(1, \hat{\lambda}_{ax} + \hat{\lambda}_{bx} - \hat{\lambda}_{abx})/m)$
  $\delta \leftarrow \varepsilon/\sqrt{m}$ ▷ $\varepsilon = 10^{-2}$
  repeat
    $\varphi'_a \leftarrow \varphi_a,\ \varphi'_b \leftarrow \varphi_b,\ \varphi'_x \leftarrow \varphi_x$
    $\varphi_a, \varphi_b, \varphi_x \leftarrow$ OptimizationStep$(\varphi_a, \varphi_b, \varphi_x; f, \nabla f)$ ▷ e.g. BFGS update step
  until $\max(|\varphi_a - \varphi'_a|, |\varphi_b - \varphi'_b|, |\varphi_x - \varphi'_x|) \leq \delta$
  $\hat{\lambda}_a \leftarrow m e^{\varphi_a},\ \hat{\lambda}_b \leftarrow m e^{\varphi_b},\ \hat{\lambda}_x \leftarrow m e^{\varphi_x}$
  return $(\hat{\lambda}_a, \hat{\lambda}_b, \hat{\lambda}_x)$
end function

lated an improvement factor which represents the root-mean-square error ratio between both approaches. Since we only observed values larger than 1, the new maximum likelihood estimation approach gives better estimates for all investigated cases. For some cases the improvement factor is clearly larger than 2. Due to square root error scaling law, this means that we would need 4 times more registers to get the same error when using the conventional approach. We also considered some cases with true cardinalities smaller than the number of registers. As the results suggest and as expected the joint estimation algorithm works well over the entire cardinality range without special handling of small cardinalities.

We also investigated, if the new approach could give better estimates for the union operation. The conventional approach merges both HyperLogLog sketches using (1) and estimates the union size using single sketch cardinality estimation. The joint cardinality estimation algorithm provides another opportunity. The union can also be estimated by simply summing up the three estimates for the sets $|A|$, $|B|$, and $|X|$. The corresponding results are shown in Table 5. As can be clearly seen the joint estimation algorithm is also able to improve the cardinality estimation of unions by a significant amount.

6. Future work

As described in Section 2.4 an unbiased estimator for the rate in the Poisson model, is also unbiased estimator for the cardinality. We have shown that this approach works well for the improved raw estimator as well as for the maximum likelihood estimator even though both are only approximately unbiased. Therefore, it would be interesting what conditions on an approximately unbiased Poisson rate estimator are sufficient to guarantee approximate unbiasedness, if used as cardinality estimator.

Using the maximum likelihood method we have been able to improve the cardinality estimates for the results of set operations between two HyperLogLog sketches. Unfortunately, joint cardinality estimation is much more expensive than for a single HyperLogLog sketch, because it requires maximization of a multi-dimensional function. Since we have found the improved raw estimator which is almost as accurate as the maximum likelihood estimator for the single HyperLogLog case, we could imagine that there also exists a faster algorithm for the two HyperLogLog case. It is expected that such a new algorithm makes use of all the information given by the sufficient statistic (70).

The presented maximum likelihood method could also be used to estimate the cardinality of set operations between more than two HyperLogLog sketches. However, further research is necessary to determine, if this is feasible from a practical point of view. As for the inclusion-exclusion principle, the effort would scale at least exponentially with the number of involved HyperLogLog sketches.

The maximum likelihood method can also be used to estimate distance measures such as the Jaccard distance of two sets that are represented as HyperLogLog sketches. This directly leads to the question whether the HyperLogLog algorithm could be used for locality-sensitive hashing [24, 25]. Various locality-sensitive hashing algorithms have been proposed in the past. Among the most popular ones are the SimHash [26] and the

Table 1: List of true cardinalities of the pairwise disjoint sets A, B, and X, for which maximum likelihood estimation from two HyperLogLog sketches with parameters $p = 20$ and $q = 44$ representing sets $S_1 = A \cup X$ and $S_2 = B \cup X$ was investigated. The last column shows the average number of iteration cycles until the stop criterion of the BFGS algorithm was satisfied.

	true cardinalities			Jaccard index	cardinality ratio	avg number
#	$\lvert A\rvert$	$\lvert B\rvert$	$\lvert X\rvert$	$\lvert S_1 \cap S_2\rvert / \lvert S_1 \cup S_2\rvert$	$\lvert S_1\rvert / \lvert S_2\rvert$	iterations
1	1 598 728 349	251 188 153	2 742 179	1.480e−3	6.365	36.206
2	13 049 219 720	5 945 604 476	103 608 860	5.425e−3	2.195	38.740
3	144 199	1608	3457	2.316e−2	89.676	25.776
4	36 085 381	8 525 744	3 516 662	7.307e−2	4.233	42.429
5	652 906 563	42 736 075	23 996 891	3.335e−2	15.278	42.131
6	74 772	2617	235	3.027e−3	28.572	22.503
7	148 568	12 597	7147	4.246e−2	11.794	30.143
8	3 837 563 445	277 467 992	104 644 949	2.480e−2	13.831	42.331
9	84 256	1624	402	4.659e−3	51.882	22.287
10	122 704 431	36 085 381	32 994 301	1.720e−1	3.400	42.104
11	22 160 727	6 987 227	586 521	1.973e−2	3.172	41.723
12	48 156 078	4 333 904	1 313 150	2.441e−2	11.112	42.233
13	184 928	117 006	27 370	8.311e−2	1.581	34.903
14	17 149	2540	1530	7.211e−2	6.752	21.743
15	19 279 012	2 939 988	873 249	3.782e−2	6.558	42.005
16	1 602 292	23 812	2277	1.398e−3	67.289	30.127
17	99 786	1923	2805	2.684e−2	51.891	25.060
18	61 756 895	744 726	974 256	1.535e−2	82.926	41.036
19	10 277 123 441	352 310 724	1 783 650 614	1.437e−1	29.171	40.823
20	134 200 029	11 958 008	15 335 333	9.496e−2	11.223	41.639
21	110 226	1205	691	6.163e−3	91.474	22.837
22	773 240 139	13 474 584	20 465 059	2.535e−2	57.385	41.597
23	307 179	5046	8549	2.665e−2	60.876	29.480
24	74 032	16 317	6343	6.560e−2	4.537	29.324
25	50 221	4898	958	1.708e−2	10.253	23.952
26	680 932	24 290	109 135	1.340e−1	28.033	35.869
27	1 097 816	452 822	48 262	3.018e−2	2.424	39.258
28	84 256	2919	2210	2.472e−2	28.865	25.251
29	13 049 219 720	721 215 037	145 319 400	1.044e−2	18.093	41.157
30	59 347 163	1 287 276	286 512	4.703e−3	46.103	39.461
31	119 095 712	5 235 826	586 521	4.695e−3	22.746	39.900
32	1 521 135 167	30 469 684	325 353 043	1.733e−1	49.923	39.982
33	1 783 650 614	21 296 023	389 170 221	1.774e−1	83.755	39.640
34	32 092	4054	272	7.469e−3	7.916	22.064
35	47 102 261 762	2 264 762 985	1 174 389 309	2.324e−2	20.798	42.338
36	15 837	13 915	1441	4.620e−2	1.138	24.239
37	3 083 083 913	1 174 389 309	162 128 255	3.668e−2	2.625	42.286
38	464	305	14	1.788e−2	1.521	13.086
39	43 163 436	1 054 980	8 441 331	1.603e−1	40.914	39.835
40	3857	3224	87	1.214e−2	1.196	17.620
41	5 329 186 306	4 114 387 435	429 886 036	4.354e−2	1.295	41.020
42	2 138 265	34 751	6343	2.910e−3	61.531	32.251
43	9 703 090	3 733 007	18 023	1.340e−3	2.599	35.449
44	721 215 037	220 709 637	6 325 446	6.671e−3	3.268	39.271
45	26 507 497	8 275 003	1 479 690	4.081e−2	3.203	42.079
46	35 805	1707	146	3.877e−3	20.975	20.768
47	750 499 260	447 341 132	3 089 958	2.573e−3	1.678	37.745
48	8 441 331	1 634 498	53 310	5.263e−3	5.164	37.912
49	5 235 826	485 486	64 405	1.113e−2	10.785	38.537
50	93 072	52 782	358	2.449e−3	1.763	26.500
51	5 172 455 725	3 208 269 483	72 414 817	8.567e−3	1.612	39.873
52	1 697 082 357	131 555 758	4 509 877	2.460e−3	12.900	37.682

Table 2: The mean, the standard deviation, and the root-mean-square error of the relative estimation error are given for all the cases given in Table 1 when estimating $|A|$ using the conventional approach and the maximum likelihood method, respectively. The root-mean-square error improvement factor of the maximum likelihood method is shown in the last column.

	conventional approach			maximum likelihood method			improvement
#	mean	stdev	rmse	mean	stdev	rmse	rmse ratio
1	2.579e−5	1.163e−3	1.164e−3	1.949e−5	1.036e−3	1.036e−3	1.123
2	−4.569e−5	1.416e−3	1.417e−3	−5.045e−6	1.087e−3	1.087e−3	1.304
3	−1.544e−5	7.116e−4	7.118e−4	−1.604e−5	7.059e−4	7.061e−4	1.008
4	−1.279e−4	1.314e−3	1.320e−3	−1.005e−4	1.104e−3	1.108e−3	1.191
5	3.850e−5	1.101e−3	1.101e−3	5.756e−5	1.043e−3	1.044e−3	1.055
6	−2.748e−5	7.267e−4	7.272e−4	−2.551e−5	7.166e−4	7.171e−4	1.014
7	−2.147e−5	7.739e−4	7.742e−4	−1.750e−5	7.338e−4	7.340e−4	1.055
8	7.057e−5	1.105e−3	1.107e−3	6.282e−5	1.041e−3	1.043e−3	1.061
9	2.594e−5	7.215e−4	7.220e−4	2.190e−5	7.150e−4	7.154e−4	1.009
10	−7.376e−6	1.461e−3	1.461e−3	−8.897e−6	1.245e−3	1.245e−3	1.173
11	−1.006e−4	1.269e−3	1.273e−3	−8.628e−5	1.031e−3	1.034e−3	1.231
12	1.130e−5	1.127e−3	1.127e−3	1.364e−5	1.028e−3	1.028e−3	1.096
13	−2.510e−5	1.169e−3	1.169e−3	−5.211e−6	9.041e−4	9.041e−4	1.293
14	−1.722e−6	8.447e−4	8.447e−4	−2.638e−5	7.680e−4	7.685e−4	1.099
15	−6.267e−5	1.152e−3	1.153e−3	−4.085e−5	1.013e−3	1.013e−3	1.138
16	−2.600e−5	7.728e−4	7.732e−4	−2.182e−5	7.643e−4	7.646e−4	1.011
17	−2.688e−5	7.380e−4	7.385e−4	−2.674e−5	7.225e−4	7.230e−4	1.021
18	1.542e−5	1.022e−3	1.022e−3	2.309e−5	1.007e−3	1.007e−3	1.014
19	−5.020e−6	1.173e−3	1.173e−3	−1.559e−5	1.087e−3	1.087e−3	1.079
20	3.797e−5	1.241e−3	1.242e−3	4.173e−5	1.140e−3	1.141e−3	1.088
21	−9.072e−6	7.129e−4	7.130e−4	−6.895e−6	7.058e−4	7.058e−4	1.010
22	1.057e−5	1.058e−3	1.058e−3	2.074e−5	1.035e−3	1.036e−3	1.022
23	1.568e−5	7.297e−4	7.299e−4	1.680e−5	7.126e−4	7.128e−4	1.024
24	−2.259e−5	8.829e−4	8.832e−4	−9.515e−6	7.850e−4	7.851e−4	1.125
25	−2.694e−5	7.876e−4	7.881e−4	−1.676e−5	7.451e−4	7.453e−4	1.057
26	−2.260e−5	8.661e−4	8.664e−4	−2.941e−5	8.191e−4	8.197e−4	1.057
27	2.699e−5	1.079e−3	1.079e−3	3.847e−5	8.342e−4	8.351e−4	1.292
28	−3.561e−5	7.604e−4	7.613e−4	−2.693e−5	7.357e−4	7.362e−4	1.034
29	1.557e−5	1.085e−3	1.085e−3	4.645e−6	1.054e−3	1.054e−3	1.030
30	1.343e−5	1.017e−3	1.017e−3	2.489e−5	9.967e−4	9.970e−4	1.020
31	2.126e−5	1.090e−3	1.091e−3	2.992e−5	1.053e−3	1.053e−3	1.036
32	−4.000e−5	1.199e−3	1.200e−3	−3.250e−5	1.114e−3	1.114e−3	1.077
33	9.949e−6	1.219e−3	1.219e−3	4.309e−6	1.141e−3	1.141e−3	1.068
34	3.038e−5	7.675e−4	7.681e−4	2.711e−5	7.201e−4	7.206e−4	1.066
35	−3.449e−5	1.082e−3	1.082e−3	−4.149e−5	1.037e−3	1.038e−3	1.043
36	−4.153e−5	1.192e−3	1.193e−3	−3.156e−5	8.941e−4	8.947e−4	1.333
37	−1.873e−6	1.385e−3	1.385e−3	3.108e−5	1.111e−3	1.112e−3	1.246
38	−3.041e−5	1.078e−3	1.078e−3	−4.340e−5	8.652e−4	8.663e−4	1.244
39	1.353e−5	1.210e−3	1.210e−3	4.388e−5	1.128e−3	1.128e−3	1.072
40	3.250e−5	1.130e−3	1.131e−3	1.258e−5	8.675e−4	8.676e−4	1.303
41	−6.626e−6	1.665e−3	1.665e−3	3.486e−5	1.181e−3	1.181e−3	1.409
42	−2.572e−5	8.165e−4	8.170e−4	−2.424e−5	8.031e−4	8.035e−4	1.017
43	−2.696e−5	1.299e−3	1.300e−3	5.629e−5	9.953e−4	9.969e−4	1.304
44	1.070e−4	1.280e−3	1.284e−3	5.107e−5	1.047e−3	1.048e−3	1.225
45	−1.509e−4	1.326e−3	1.335e−3	−1.016e−4	1.070e−3	1.075e−3	1.241
46	−9.710e−6	7.332e−4	7.332e−4	−9.042e−6	7.144e−4	7.145e−4	1.026
47	1.586e−5	1.475e−3	1.475e−3	4.384e−5	1.062e−3	1.063e−3	1.389
48	−3.650e−5	1.144e−3	1.144e−3	3.333e−6	9.695e−4	9.695e−4	1.180
49	−4.563e−6	9.927e−4	9.927e−4	1.056e−5	9.255e−4	9.255e−4	1.073
50	2.791e−5	1.034e−3	1.034e−3	−2.092e−5	7.978e−4	7.981e−4	1.296
51	−1.885e−5	1.529e−3	1.529e−3	1.499e−5	1.098e−3	1.098e−3	1.392
52	3.343e−5	1.094e−3	1.094e−3	1.743e−5	1.016e−3	1.016e−3	1.077

Table 3: The mean, the standard deviation, and the root-mean-square error of the relative estimation error are given for all the cases given in Table 1 when estimating $|B|$ using the conventional approach and the maximum likelihood method, respectively. The root-mean-square error improvement factor of the maximum likelihood method is shown in the last column.

	conventional approach			maximum likelihood method			improvement
#	mean	stdev	rmse	mean	stdev	rmse	rmse ratio
1	6.070e−5	3.698e−3	3.698e−3	2.165e−5	1.576e−3	1.576e−3	2.346
2	−9.405e−5	2.300e−3	2.302e−3	−5.921e−6	1.216e−3	1.216e−3	1.894
3	−9.735e−5	9.642e−3	9.643e−3	−1.821e−5	7.152e−3	7.152e−3	1.348
4	−1.812e−4	3.294e−3	3.299e−3	−9.105e−5	2.021e−3	2.023e−3	1.631
5	−3.106e−4	5.731e−3	5.739e−3	−5.314e−5	3.482e−3	3.482e−3	1.648
6	−1.113e−4	5.239e−3	5.240e−3	−4.500e−5	3.184e−3	3.184e−3	1.646
7	−7.503e−5	3.540e−3	3.541e−3	−2.983e−5	2.452e−3	2.453e−3	1.444
8	1.721e−4	5.479e−3	5.482e−3	9.461e−5	3.132e−3	3.133e−3	1.749
9	2.680e−4	7.139e−3	7.144e−3	8.318e−5	4.489e−3	4.489e−3	1.591
10	6.864e−5	3.124e−3	3.125e−3	7.575e−5	2.216e−3	2.217e−3	1.410
11	−1.117e−4	2.732e−3	2.734e−3	−7.133e−5	1.452e−3	1.454e−3	1.881
12	5.305e−5	5.045e−3	5.045e−3	1.053e−5	2.734e−3	2.734e−3	1.845
13	4.409e−5	1.546e−3	1.546e−3	7.546e−5	1.108e−3	1.111e−3	1.392
14	−3.640e−5	2.754e−3	2.754e−3	−1.429e−4	1.916e−3	1.921e−3	1.434
15	−7.580e−5	3.874e−3	3.875e−3	4.206e−5	2.185e−3	2.186e−3	1.773
16	−4.068e−4	9.791e−3	9.799e−3	−1.341e−4	4.726e−3	4.728e−3	2.073
17	−2.882e−4	7.282e−3	7.287e−3	−2.741e−4	5.387e−3	5.394e−3	1.351
18	2.041e−4	1.335e−2	1.335e−2	1.917e−4	8.984e−3	8.987e−3	1.485
19	1.513e−4	8.079e−3	8.081e−3	−1.609e−4	6.349e−3	6.351e−3	1.272
20	−1.084e−4	5.134e−3	5.135e−3	−3.238e−5	3.587e−3	3.587e−3	1.432
21	4.438e−5	9.444e−3	9.444e−3	1.792e−4	6.519e−3	6.522e−3	1.448
22	1.913e−4	1.069e−2	1.069e−2	3.021e−4	7.523e−3	7.529e−3	1.420
23	−2.136e−4	8.114e−3	8.117e−3	−1.427e−4	5.903e−3	5.905e−3	1.375
24	−4.482e−5	2.316e−3	2.316e−3	3.233e−6	1.617e−3	1.617e−3	1.433
25	−6.055e−5	3.229e−3	3.229e−3	4.118e−5	2.058e−3	2.058e−3	1.569
26	4.871e−4	6.121e−3	6.140e−3	2.760e−4	4.699e−3	4.707e−3	1.304
27	9.544e−5	1.980e−3	1.982e−3	1.260e−4	1.195e−3	1.202e−3	1.649
28	−4.552e−4	5.557e−3	5.575e−3	−2.381e−4	3.899e−3	3.906e−3	1.427
29	6.400e−5	6.155e−3	6.155e−3	−8.843e−5	3.201e−3	3.203e−3	1.922
30	−5.574e−4	9.629e−3	9.645e−3	−1.373e−4	4.880e−3	4.882e−3	1.976
31	−2.529e−4	7.028e−3	7.033e−3	−7.499e−5	3.126e−3	3.127e−3	2.249
32	−1.911e−5	1.114e−2	1.114e−2	7.799e−5	8.776e−3	8.776e−3	1.270
33	6.968e−4	1.463e−2	1.465e−2	−1.102e−4	1.162e−2	1.162e−2	1.261
34	−6.731e−5	2.863e−3	2.864e−3	−9.074e−5	1.807e−3	1.810e−3	1.583
35	3.453e−5	6.839e−3	6.839e−3	−4.967e−5	3.976e−3	3.976e−3	1.720
36	−6.295e−6	1.305e−3	1.305e−3	8.557e−6	9.825e−4	9.825e−4	1.328
37	−8.657e−5	2.537e−3	2.538e−3	−7.011e−6	1.502e−3	1.502e−3	1.690
38	−1.139e−5	1.388e−3	1.388e−3	−3.061e−5	9.995e−4	9.999e−4	1.388
39	−5.702e−4	1.010e−2	1.012e−2	−4.754e−4	7.874e−3	7.889e−3	1.282
40	3.115e−5	1.272e−3	1.273e−3	6.544e−6	9.479e−4	9.479e−4	1.343
41	4.213e−5	1.958e−3	1.959e−3	9.613e−5	1.278e−3	1.281e−3	1.529
42	−9.922e−5	9.654e−3	9.654e−3	−3.566e−5	5.001e−3	5.001e−3	1.930
43	−6.338e−5	2.479e−3	2.480e−3	1.526e−4	1.208e−3	1.218e−3	2.037
44	2.256e−4	2.816e−3	2.825e−3	4.582e−5	1.365e−3	1.365e−3	2.069
45	−1.837e−4	2.869e−3	2.875e−3	−4.060e−5	1.627e−3	1.628e−3	1.766
46	5.711e−6	4.558e−3	4.558e−3	2.898e−5	2.769e−3	2.769e−3	1.646
47	−7.831e−5	2.127e−3	2.128e−3	−3.118e−5	1.167e−3	1.168e−3	1.822
48	−1.312e−4	3.327e−3	3.330e−3	7.075e−5	1.407e−3	1.408e−3	2.364
49	−1.350e−4	4.579e−3	4.581e−3	1.869e−5	2.176e−3	2.176e−3	2.105
50	3.961e−5	1.506e−3	1.507e−3	−4.623e−5	9.511e−4	9.522e−4	1.582
51	−3.662e−5	2.086e−3	2.086e−3	1.828e−5	1.204e−3	1.204e−3	1.733
52	1.945e−4	5.199e−3	5.203e−3	−6.700e−6	2.075e−3	2.075e−3	2.507

Table 4: The mean, the standard deviation, and the root-mean-square error of the relative estimation error are given for all the cases given in Table 1 when estimating $|X|$ using the conventional approach and the maximum likelihood method, respectively. The root-mean-square error improvement factor of the maximum likelihood method is shown in the last column.

	conventional approach			maximum likelihood method			improvement
#	mean	stdev	rmse	mean	stdev	rmse	rmse ratio
1	−4.875e−3	3.264e−1	3.264e−1	−1.494e−3	1.116e−1	1.116e−1	2.926
2	7.323e−3	1.195e−1	1.198e−1	2.263e−3	4.011e−2	4.017e−2	2.982
3	8.163e−5	4.503e−3	4.504e−3	4.620e−5	3.369e−3	3.370e−3	1.337
4	3.306e−4	7.642e−3	7.649e−3	1.126e−4	4.398e−3	4.399e−3	1.739
5	4.956e−4	1.008e−2	1.009e−2	3.823e−5	6.060e−3	6.060e−3	1.666
6	7.947e−4	5.767e−2	5.767e−2	6.096e−5	3.455e−2	3.455e−2	1.669
7	1.584e−4	5.994e−3	5.996e−3	8.149e−5	4.109e−3	4.110e−3	1.459
8	−3.991e−4	1.427e−2	1.428e−2	−1.940e−4	7.872e−3	7.874e−3	1.813
9	−1.141e−3	2.867e−2	2.869e−2	−3.961e−4	1.788e−2	1.788e−2	1.604
10	−3.401e−6	3.211e−3	3.211e−3	−9.036e−6	2.292e−3	2.292e−3	1.401
11	7.400e−4	3.069e−2	3.070e−2	2.564e−4	1.357e−2	1.357e−2	2.262
12	−2.408e−4	1.631e−2	1.631e−2	−1.030e−4	8.539e−3	8.539e−3	1.910
13	−1.033e−4	5.661e−3	5.662e−3	−2.275e−4	3.602e−3	3.610e−3	1.568
14	8.121e−6	4.318e−3	4.318e−3	1.841e−4	2.994e−3	2.999e−3	1.440
15	3.446e−5	1.267e−2	1.267e−2	−3.619e−4	6.869e−3	6.878e−3	1.841
16	4.114e−3	1.020e−1	1.020e−1	1.261e−3	4.868e−2	4.869e−2	2.096
17	2.230e−4	4.982e−3	4.987e−3	2.135e−4	3.722e−3	3.728e−3	1.338
18	−1.960e−4	1.022e−2	1.022e−2	−1.868e−4	6.876e−3	6.878e−3	1.486
19	3.677e−5	1.853e−3	1.854e−3	9.680e−5	1.565e−3	1.568e−3	1.182
20	6.430e−5	4.009e−3	4.010e−3	6.464e−6	2.844e−3	2.844e−3	1.410
21	−1.583e−4	1.641e−2	1.641e−2	−3.933e−4	1.129e−2	1.130e−2	1.452
22	−2.769e−5	7.100e−3	7.100e−3	−1.008e−4	5.031e−3	5.032e−3	1.411
23	1.567e−4	4.724e−3	4.726e−3	1.157e−4	3.478e−3	3.480e−3	1.358
24	8.273e−5	5.475e−3	5.476e−3	−3.789e−5	3.675e−3	3.676e−3	1.490
25	3.269e−4	1.614e−2	1.615e−2	−1.856e−4	9.953e−3	9.954e−3	1.622
26	−1.456e−4	1.455e−3	1.462e−3	−1.034e−4	1.222e−3	1.226e−3	1.193
27	−1.100e−3	1.715e−2	1.719e−2	−1.375e−3	8.919e−3	9.024e−3	1.905
28	5.458e−4	7.245e−3	7.266e−3	2.645e−4	5.064e−3	5.071e−3	1.433
29	−3.501e−4	3.005e−2	3.005e−2	4.066e−4	1.500e−2	1.501e−2	2.002
30	2.367e−3	4.322e−2	4.329e−2	4.796e−4	2.168e−2	2.168e−2	1.996
31	2.328e−3	6.227e−2	6.231e−2	7.398e−4	2.688e−2	2.689e−2	2.317
32	−4.899e−5	1.430e−3	1.431e−3	−5.790e−5	1.294e−3	1.295e−3	1.104
33	−8.169e−5	1.273e−3	1.275e−3	−3.903e−5	1.185e−3	1.185e−3	1.076
34	3.101e−4	4.123e−2	4.123e−2	6.603e−4	2.465e−2	2.466e−2	1.672
35	−6.657e−5	1.308e−2	1.308e−2	9.568e−5	7.445e−3	7.446e−3	1.756
36	−7.708e−6	1.020e−2	1.020e−2	−1.295e−4	6.264e−3	6.266e−3	1.629
37	7.793e−4	1.699e−2	1.701e−2	2.039e−4	8.106e−3	8.109e−3	2.097
38	1.524e−4	2.619e−2	2.619e−2	5.635e−4	1.552e−2	1.553e−2	1.687
39	5.109e−5	1.557e−3	1.557e−3	3.882e−5	1.357e−3	1.358e−3	1.147
40	−1.435e−3	3.909e−2	3.912e−2	−5.471e−4	2.338e−2	2.339e−2	1.672
41	2.821e−4	1.590e−2	1.590e−2	−2.253e−4	7.547e−3	7.550e−3	2.106
42	6.015e−4	5.277e−2	5.277e−2	2.526e−4	2.709e−2	2.709e−2	1.948
43	2.159e−2	4.683e−1	4.688e−1	−2.983e−2	1.733e−1	1.758e−1	2.666
44	−8.114e−3	9.074e−2	9.110e−2	−1.840e−3	3.189e−2	3.194e−2	2.852
45	6.959e−4	1.502e−2	1.503e−2	−1.025e−4	7.428e−3	7.429e−3	2.024
46	−5.473e−5	5.257e−2	5.257e−2	−3.208e−4	3.123e−2	3.124e−2	1.683
47	1.150e−2	2.669e−1	2.672e−1	4.706e−3	8.069e−2	8.083e−2	3.306
48	4.006e−3	9.917e−2	9.925e−2	−2.182e−3	3.597e−2	3.603e−2	2.755
49	6.802e−4	3.392e−2	3.393e−2	−4.765e−4	1.543e−2	1.544e−2	2.197
50	−7.867e−3	1.953e−1	1.955e−1	4.774e−3	9.479e−2	9.491e−2	2.060
51	3.506e−3	8.148e−2	8.155e−2	1.085e−3	2.942e−2	2.944e−2	2.770
52	−5.209e−3	1.488e−1	1.489e−1	6.617e−4	5.339e−2	5.340e−2	2.788

Table 5: The mean, the standard deviation, and the root-mean-square error of the relative estimation error are given for all the cases given in Table 1 when estimating $|A \cup B \cup X|$ using the conventional approach and the maximum likelihood method, respectively. The root-mean-square error improvement factor of the maximum likelihood method is shown in the last column.

	conventional approach			maximum likelihood method			improvement
#	mean	stdev	rmse	mean	stdev	rmse	rmse ratio
1	2.326e−5	1.012e−3	1.012e−3	1.754e−5	9.033e−4	9.035e−4	1.121
2	−2.077e−5	1.016e−3	1.016e−3	6.986e−6	8.070e−4	8.070e−4	1.259
3	−1.407e−5	6.875e−4	6.876e−4	−1.462e−5	6.819e−4	6.821e−4	1.008
4	−1.038e−4	1.015e−3	1.020e−3	−8.322e−5	8.601e−4	8.641e−4	1.180
5	3.301e−5	1.004e−3	1.004e−3	5.034e−5	9.508e−4	9.521e−4	1.055
6	−2.782e−5	7.006e−4	7.012e−4	−2.590e−5	6.908e−4	6.913e−4	1.014
7	−1.784e−5	6.860e−4	6.862e−4	−1.422e−5	6.492e−4	6.494e−4	1.057
8	6.560e−5	1.009e−3	1.011e−3	5.854e−5	9.513e−4	9.531e−4	1.061
9	2.506e−5	7.044e−4	7.049e−4	2.111e−5	6.981e−4	6.984e−4	1.009
10	7.610e−6	1.010e−3	1.010e−3	7.007e−6	8.690e−4	8.690e−4	1.162
11	−8.663e−5	9.706e−4	9.745e−4	−7.600e−5	7.978e−4	8.014e−4	1.216
12	8.512e−6	1.013e−3	1.013e−3	1.054e−5	9.234e−4	9.235e−4	1.097
13	−7.018e−6	7.193e−4	7.194e−4	4.975e−6	5.749e−4	5.750e−4	1.251
14	−5.163e−6	6.996e−4	6.997e−4	−2.515e−5	6.345e−4	6.350e−4	1.102
15	−6.067e−5	9.693e−4	9.712e−4	−4.243e−5	8.516e−4	8.527e−4	1.139
16	−2.578e−5	7.605e−4	7.609e−4	−2.167e−5	7.522e−4	7.525e−4	1.011
17	−2.498e−5	7.063e−4	7.068e−4	−2.484e−5	6.914e−4	6.918e−4	1.022
18	1.439e−5	9.942e−4	9.943e−4	2.185e−5	9.801e−4	9.803e−4	1.014
19	5.421e−6	9.911e−4	9.912e−4	−3.562e−6	9.221e−4	9.221e−4	1.075
20	2.963e−5	1.045e−3	1.045e−3	3.289e−5	9.606e−4	9.612e−4	1.088
21	−9.417e−6	7.007e−4	7.008e−4	−7.277e−6	6.937e−4	6.937e−4	1.010
22	1.262e−5	1.014e−3	1.014e−3	2.235e−5	9.928e−4	9.930e−4	1.022
23	1.583e−5	6.998e−4	7.000e−4	1.693e−5	6.831e−4	6.833e−4	1.024
24	−1.943e−5	6.940e−4	6.943e−4	−9.225e−6	6.153e−4	6.154e−4	1.128
25	−2.383e−5	7.081e−4	7.085e−4	−1.458e−5	6.702e−4	6.704e−4	1.057
26	−2.388e−5	7.319e−4	7.322e−4	−3.022e−5	6.908e−4	6.914e−4	1.059
27	1.237e−5	7.709e−4	7.710e−4	2.060e−5	6.079e−4	6.082e−4	1.268
28	−3.494e−5	7.191e−4	7.200e−4	−2.662e−5	6.957e−4	6.962e−4	1.034
29	1.426e−5	1.020e−3	1.020e−3	4.019e−6	9.907e−4	9.907e−4	1.030
30	1.243e−5	9.910e−4	9.911e−4	2.360e−5	9.714e−4	9.717e−4	1.020
31	2.060e−5	1.041e−3	1.041e−3	2.886e−5	1.005e−3	1.005e−3	1.035
32	−4.122e−5	9.892e−4	9.901e−4	−3.511e−5	9.211e−4	9.217e−4	1.074
33	3.613e−7	1.007e−3	1.007e−3	−4.490e−6	9.463e−4	9.463e−4	1.064
34	2.160e−5	6.817e−4	6.821e−4	1.872e−5	6.402e−4	6.405e−4	1.065
35	−3.214e−5	1.012e−3	1.012e−3	−3.867e−5	9.703e−4	9.711e−4	1.042
36	−2.425e−5	7.009e−4	7.014e−4	−1.819e−5	5.681e−4	5.684e−4	1.234
37	4.276e−6	1.008e−3	1.008e−3	2.730e−5	8.299e−4	8.304e−4	1.214
38	−1.973e−5	6.893e−4	6.896e−4	−2.757e−5	5.761e−4	5.767e−4	1.196
39	7.854e−6	1.007e−3	1.007e−3	3.266e−5	9.397e−4	9.402e−4	1.071
40	1.409e−5	6.907e−4	6.909e−4	3.071e−6	5.692e−4	5.693e−4	1.214
41	2.626e−5	1.004e−3	1.005e−3	4.906e−5	7.722e−4	7.737e−4	1.298
42	−2.506e−5	8.009e−4	8.013e−4	−2.362e−5	7.878e−4	7.881e−4	1.017
43	−8.101e−6	9.741e−4	9.741e−4	4.299e−5	7.492e−4	7.504e−4	1.298
44	7.979e−5	1.009e−3	1.012e−3	3.723e−5	8.350e−4	8.358e−4	1.211
45	−1.239e−4	1.001e−3	1.008e−3	−8.769e−5	8.165e−4	8.212e−4	1.228
46	−9.186e−6	6.975e−4	6.975e−4	−8.528e−6	6.796e−4	6.797e−4	1.026
47	1.035e−5	1.002e−3	1.002e−3	2.789e−5	7.646e−4	7.651e−4	1.310
48	−3.050e−5	9.625e−4	9.630e−4	2.710e−6	8.188e−4	8.188e−4	1.176
49	−7.883e−6	9.005e−4	9.006e−4	5.823e−6	8.391e−4	8.391e−4	1.073
50	1.280e−5	7.030e−4	7.032e−4	−1.831e−5	5.621e−4	5.624e−4	1.250
51	4.600e−6	1.005e−3	1.005e−3	2.540e−5	7.701e−4	7.705e−4	1.305
52	3.210e−5	1.014e−3	1.015e−3	1.728e−5	9.420e−4	9.422e−4	1.077

minwise hashing [27] algorithms whose hash collision probabilities are a function of the angular and Jaccard distances, respectively. A generalization of the latter method is b-bit minwise hashing that improves memory efficiency by only storing the lowest b bits of the minimum hash value [28]. The probability that two different sets are mapped to equal hash values K_1 and K_2 is roughly

$$P(K_1 = K_2) \approx 1 - (1 - \tfrac{1}{2^b})D. \tag{75}$$

The HyperLogLog algorithm itself can be regarded as hashing algorithm as it maps sets to register values. For sufficiently large cardinalities we can use the Poisson approximation and assume that the number of zero-valued HyperLogLog registers can be ignored. Furthermore, if the HyperLogLog parameter q is chosen large enough, the number of saturated registers can be ignored as well. As a consequence, we can assume that the distribution of register values follows (17) and the probability that a register has the same value for two different sets is (compare (65))

$$P(K_1 = K_2) = \sum_{k=-\infty}^{\infty} e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^k}} \left(1 - e^{-\frac{\lambda_a+\lambda_x}{m2^k}} - e^{-\frac{\lambda_b+\lambda_x}{m2^k}} + e^{-\frac{\lambda_a+\lambda_b+\lambda_x}{m2^k}}\right). \tag{76}$$

Using the approximation $\sum_{k=-\infty}^{\infty} e^{-\frac{x}{2^k}} - e^{-\frac{y}{2^k}} \approx 2\alpha_\infty \left(\log(y) - \log(x)\right)$ (compare (89) in Appendix A) we get

$$P(K_1 = K_2) \approx 1 + 2\alpha_\infty \log\Big(1 - \tfrac{1}{2}D + \tfrac{1}{4}D^2 \tfrac{\lambda_a\lambda_b}{(\lambda_a+\lambda_b)^2}\Big) \tag{77}$$

where $D = \frac{\lambda_a+\lambda_b}{\lambda_a+\lambda_b+\lambda_x}$ is the Jaccard distance. Since $\frac{\lambda_a\lambda_b}{(\lambda_a+\lambda_b)^2}$ is always in the range $[0, \frac{1}{4}]$, the probability for equal register values can be bounded by

$$1 + 2\alpha_\infty \log\big(1 - \tfrac{1}{2}D\big) \lesssim P(K_1 = K_2) \lesssim 1 + 2\alpha_\infty \log\big(1 - \tfrac{1}{2}D + \tfrac{1}{16}D^2\big). \tag{78}$$

As shown in Fig. 21 the bounds are very close, especially for small Jaccard distances, where the probability can be well approximated by

$$P(K_1 = K_2) \approx 1 - \alpha_\infty D. \tag{79}$$

This dependency on the Jaccard distance is very similar to that of minwise hashing (75), which makes the HyperLogLog algorithm an interesting candidate for locality-sensitive hashing with respect to the Jaccard similarity. The memory-efficiency of different hashing approaches can be measured by a storage factor that is the variance of the distance estimator multiplied by the number of bits used for storing the hash signature [28]. For a hash algorithm that maps two different sets to the same b-bit hash value with probability $1 - aD$ with some constant $a \in (0, 1]$, the storage factor is given by $bD\left(\frac{1}{a} - D\right)$. Particularly, for the case $D = 0.3$, this gives 6.72 for conventional minwise hashing with 32-bit hash values ($b = 32$, $a \approx 1$), 0.51 for 1-bit minwise hashing ($b = 1$, $a = \frac{1}{2}$), and 1.63 for the HyperLogLog algorithm with 5-bit registers ($b = 5$, $a = \alpha_\infty$). This means that the HyperLogLog algorithm is not as memory-efficient as 1-bit minwise hashing,

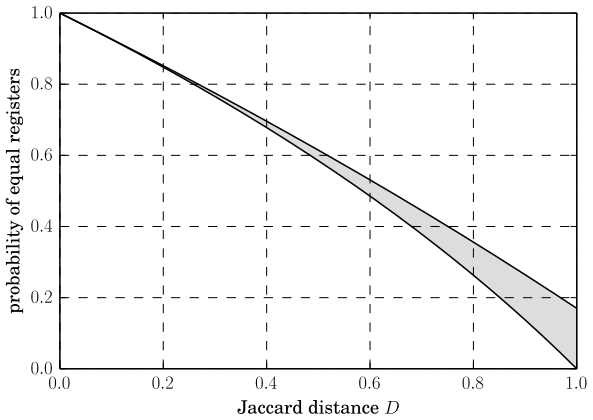

Figure 21: The approximate probability range of equal register values as a function of the Jaccard distance.

but significantly better than 32-bit minwise hashing, when estimating the Jaccard distance by just counting equal registers. However, in contrast to 1-bit minwise hashing the HyperLogLog sketch contains additional information which allows estimating the cardinality or merging two HyperLogLog hash signatures. Furthermore, the method described in Section 5 would allow a more accurate estimation of the Jaccard distance by using the estimates for intersection and union sizes. This could be used for additional more precise filtering when searching for similar items.

Since HyperLogLog sketches can be effectively constructed, because only a single hash function evaluation is needed for each item, the preprocessing step would be very fast. In contrast, preprocessing is very costly for minwise hashing, because of the many required permutations [28]. To overcome this problem, one permutation hashing was proposed [29] which keeps for a predefined number of bins the minimum of all values that are mapped to this bin. Actually, this is very similar to the HyperLogLog algorithm. The bins correspond to HyperLogLog registers which keep the first one-bit position of the minimum value instead of the minimum value itself. This close connection was also pointed out in [11], where both methods have been considered as variants of k-partion minwise hashing. The only difference is that the HyperLogLog algorithm uses base-2 ranks while one permutation hashing uses full ranks.

7. Conclusion

We have presented new algorithms for the estimation of cardinalities from HyperLogLog sketches based on the Poisson approximation. For the estimation from a single sketch, we have developed two fast algorithms that make use of the Poisson approximation. The first uses the original estimator extended by theoretically motivated correction terms. The second is based on the maximum likelihood method and solves the corresponding optimization problem using the secant method. Both algorithms are inherently unbiased

and can be implemented without dependence on empirically determined bias correction data.

The maximum likelihood method was also applied to the estimation of set operation result sizes where the operands are represented by HyperLogLog sketches. The new approach improves the cardinality estimates of set intersections, relative complements, as well as unions significantly when compared to conventional approaches such as the inclusion-exclusion principle.

A. Analysis of $\xi(x)$

The Fourier series of the periodic function

$$\xi(x) := \log(2) \sum_{k=-\infty}^{\infty} 2^{k+x} e^{-2^{k+x}}, \tag{80}$$

which has a period equal to 1, is

$$\xi(x) = \frac{a_0}{2} + \mathrm{Re}\left(\sum_{l=1}^{\infty} a_l e^{2\pi \mathrm{i} l x} \right) \tag{81}$$

with coefficients

$$\begin{aligned} a_l = 2 \int_0^1 \xi(x) e^{-2\pi \mathrm{i} l x} dx = 2\log(2) \int_0^1 \sum_{k=-\infty}^{\infty} 2^{k+x} e^{-2^{k+x}} e^{-2\pi \mathrm{i} l x} dx = \\ = 2\log(2) \sum_{k=-\infty}^{\infty} \int_k^{k+1} 2^x e^{-2^x} e^{-2\pi \mathrm{i} l x} dx = 2\log(2) \int_{-\infty}^{\infty} 2^x e^{-2^x} e^{-2\pi \mathrm{i} l x} dx. \end{aligned} \tag{82}$$

The variable transformation $y = 2^x$ yields

$$a_l = 2 \int_0^{\infty} e^{-y} y^{-\frac{2\pi \mathrm{i} l}{\log(2)}} dy = 2\Gamma\left(1 - \frac{2\pi \mathrm{i} l}{\log(2)}\right) \tag{83}$$

where Γ denotes the gamma function. Obviously, the constant term in the Fourier series is equal to 1, because $a_0 = 2$.

In order to investigate the maximum deviation of $\xi(x)$ from this value we consider all further coefficients a_l with $l \geq 1$. Using the identity $|\Gamma(1 + \mathrm{i}x)| = \sqrt{\frac{\pi x}{\sinh(\pi x)}}$ we are able to write for the absolute values of the coefficients

$$|a_l| = 2\sqrt{\frac{bl}{\sinh(bl)}} \quad \text{with } b := \frac{2\pi^2}{\log 2}. \tag{84}$$

In particular, the amplitude of the first harmonic is $|a_1| \approx 9.884 \times 10^{-6}$. Clearly, the maximum deviation of $\xi(x) - 1$ from the first harmonic must be smaller than $\sum_{l=2}^{\infty} |a_l|$.

The ratio of subsequent coefficients is given by

$$\frac{|a_{l+1}|}{|a_l|} = \sqrt{\frac{l+1}{l}}\sqrt{\frac{\sinh(bl)}{\sinh(b(l+1))}} = \sqrt{\frac{l+1}{l}}\sqrt{\frac{1}{\cosh(b)+\frac{\sinh(b)}{\tanh(bl)}}}. \tag{85}$$

For $l \geq 2$ we have $\sqrt{\frac{l+1}{l}} \leq \sqrt{\frac{3}{2}}$. Together with $\tanh(x) \leq 1$ we obtain

$$\frac{|a_{l+1}|}{|a_l|} \leq \sqrt{\frac{3}{2}}\sqrt{\frac{1}{\cosh(b)+\sinh(b)}} = \sqrt{\frac{3}{2e^b}} \tag{86}$$

which leads to $|a_l| \leq |a_2|\left(\sqrt{\frac{3}{2e^b}}\right)^{l-2}$ for $l \geq 2$ and further to

$$\sum_{l=2}^{\infty}|a_l| \leq |a_2|\sum_{l=0}^{\infty}\left(\sqrt{\frac{3}{2e^b}}\right)^{l} = 2\sqrt{\frac{2b}{\sinh(2b)}}\frac{1}{1-\sqrt{\frac{3}{2e^b}}} \approx 9.154\times 10^{-12}. \tag{87}$$

As a consequence, the maximum deviation of $\xi(x)$ from 1 is bounded by

$$9.884\times 10^{-6} \leq |a_1| - \sum_{l=2}^{\infty}|a_l| \leq \max_x(|\xi(x)-1|) \leq |a_1| + \sum_{l=2}^{\infty}|a_l| \leq 9.885\times 10^{-6}. \tag{88}$$

An interesting approximation formula can be derived from $\xi(x) \approx 1$ by integrating on both sides:

$$\sum_{k=-\infty}^{\infty} e^{-2^{k+y}} - e^{-2^{k+x}} \approx x - y. \tag{89}$$

B. Numerical stability of recursion formula for $h(x)$

In order to investigate the error propagation of a single recursion step using (55) we define $h_1 := h(2x)$ and $h_2 := h(4x)$. The recursion formula simplifies to

$$h_2 = \frac{x + h_1(1-h_1)}{x+(1-h_1)}. \tag{90}$$

If h_1 is approximated by $\tilde{h}_1 = h_1(1+\varepsilon_1)$ with relative error ε_1, the recursion formula will give an approximation for h_2

$$\tilde{h}_2 = \frac{x + \tilde{h}_1(1-\tilde{h}_1)}{x+(1-\tilde{h}_1)}. \tag{91}$$

The corresponding relative error ε_2 is given by

$$\varepsilon_2 = \frac{\tilde{h}_2}{h_2} - 1. \tag{92}$$

Combination of (90), (91), and (92) yields for its absolute value

$$|\varepsilon_2| = |\varepsilon_1| \frac{\left| \frac{h_1(1-2h_1)}{x+h_1(1-h_1)} + \frac{h_1}{x+1-h_1} - \varepsilon_1 \frac{h_1^2}{x+h_1(1-h_1)} \right|}{\left| 1 - \varepsilon_1 \frac{h_1}{x+1-h_1} \right|} \tag{93}$$

and the triangle inequality leads to

$$|\varepsilon_2| \leq |\varepsilon_1| \frac{\left| \frac{h_1(1-2h_1)}{x+h_1(1-h_1)} + \frac{h_1}{x+1-h_1} \right| + |\varepsilon_1| \frac{h_1^2}{x+h_1(1-h_1)}}{\left| 1 - \varepsilon_1 \frac{h_1}{x+1-h_1} \right|}. \tag{94}$$

By numerical means it is easy to show that the inequalities $\left| \frac{h_1(1-2h_1)}{x+h_1(1-h_1)} + \frac{h_1}{x+1-h_1} \right| \leq 0.517$, $\frac{h_1^2}{x+h_1(1-h_1)} \leq 0.436$, and $\frac{h_1}{x+1-h_1} \leq 0.529$ hold for all $x \geq 0$. Therefore, if we additionally assume, for example, $|\varepsilon_1| \leq 0.1$, we get

$$|\varepsilon_2| \leq |\varepsilon_1| \frac{0.517 + 0.1 \cdot 0.436}{1 - 0.1 \cdot 0.529} \leq |\varepsilon_1| \cdot 0.592, \tag{95}$$

which means that the relative error is decreasing in each recursion step and the recursive calculation of h is numerically stable.

C. Error caused by approximation of $h(x)$

According to (41) the exact estimate $\hat{x}$ fulfills

$$\hat{x} \sum_{k=0}^{q} \frac{C_k}{2^k} + \sum_{k=1}^{q} C_k h\left(\frac{\hat{x}}{2^k} \right) + C_{q+1} h\left(\frac{\hat{x}}{2^q} \right) = m - C_0. \tag{96}$$

If h is not calculated exactly but approximated by $\tilde{h}$ with maximum relative error $\varepsilon_h \ll 1$

$$\left| \tilde{h}(x) - h(x) \right| \leq \varepsilon_h h(x) \tag{97}$$

the solution of the equation will be off by some relative error ε_x:

$$\hat{x}\,(1+\varepsilon_x) \sum_{k=0}^{q} \frac{C_k}{2^k} + \sum_{k=1}^{q} C_k \tilde{h}\left(\frac{\hat{x}\,(1+\varepsilon_x)}{2^k} \right) + C_{q+1} \tilde{h}\left(\frac{\hat{x}\,(1+\varepsilon_x)}{2^q} \right) = m - C_0. \tag{98}$$

Due to (97) there exists some $\alpha \in [-\varepsilon_h, \varepsilon_h]$ for which

$$\hat{x}\,(1+\varepsilon_x) \sum_{k=0}^{q} \frac{C_k}{2^k} + (1+\alpha) \sum_{k=1}^{q} C_k h\left(\frac{\hat{x}\,(1+\varepsilon_x)}{2^k} \right) + \\ + (1+\alpha)\, C_{q+1} h\left(\frac{\hat{x}\,(1+\varepsilon_x)}{2^q} \right) = m - C_0. \tag{99}$$

Since $h'(x) \in [0, 0.5]$ for $x \geq 0$, there exists a $\beta \in [0, 0.5]$ for which

$$\hat{x}\,(1+\varepsilon_x)\sum_{k=0}^{q}\frac{C_k}{2^k} + (1+\alpha)\left(\sum_{k=1}^{q} C_k h\left(\frac{\hat{x}}{2^k}\right) + \frac{C_k}{2^k}\hat{x}\varepsilon_x\beta\right) + \\ + (1+\alpha)\left(C_{q+1}h\left(\frac{\hat{x}}{2^q}\right) + \frac{C_{q+1}}{2^q}\hat{x}\varepsilon_x\beta\right) = m - C_0. \quad (100)$$

Subtracting (96) multiplied by $(1+\alpha)$ from (100) and resolving ε_x gives

$$\varepsilon_x = \alpha\frac{\hat{x}\sum_{k=0}^{q}\frac{C_k}{2^k} - (m - C_0)}{\hat{x}\left(\sum_{k=0}^{q}\frac{C_k}{2^k} + (1+\alpha)\,\beta\left(\sum_{k=1}^{q}\frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}\right)\right)}. \quad (101)$$

Using $|\alpha| \leq \varepsilon_h$, $\beta \geq 0$, and (49) the absolute value of the relative error can be bounded by

$$|\varepsilon_x| \leq |\varepsilon_h|\,\frac{(m - C_0) - \hat{x}\sum_{k=0}^{q}\frac{C_k}{2^k}}{\hat{x}\sum_{k=0}^{q}\frac{C_k}{2^k}}. \quad (102)$$

Furthermore, using (46) we finally get

$$|\varepsilon_x| \leq |\varepsilon_h|\,\frac{\frac{1}{2}\sum_{k=1}^{q}\frac{C_k}{2^k} + \frac{C_{q+1}}{2^q}}{\sum_{k=0}^{q}\frac{C_k}{2^k}} \leq |\varepsilon_h|\left(\frac{1}{2} + \frac{\frac{C_{q+1}}{2^q}}{\sum_{k=0}^{q}\frac{C_k}{2^k}}\right) \leq |\varepsilon_h|\left(\frac{1}{2} + \frac{C_{q+1}}{m - C_{q+1}}\right). \quad (103)$$

Hence, as long as most registers are not saturated ($C_{q+1} \ll m$), the relative error ε_x of the calculated estimate using the approximation $\tilde{h}(x)$ for $h(x)$ has the same order of magnitude as ε_h.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137 – 147, 1999.

[2] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Why go logarithmic if we can go linear?: Towards effective distinct counting of search traffic. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, pages 618–629, Nantes, France, March 2008.

[3] Daniel Ting. Streamed approximate counting of distinct elements: Beating optimal batch methods. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 442–451, New York, NY, USA, August 2014.

[4] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 283–288, Cambridge, MA, USA, October 2003.

[5] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 41–52, Indianapolis, IN, USA, June 2010.

[6] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the 13th Conference on Analysis of Algorithms*, pages 127–146, Juan des Pins, France, June 2007.

[7] Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 683–692, Genoa, Italy, March 2013.

[8] Lee Rhodes. System and method for enhanced accuracy cardinality estimation, September 24 2015. US Patent 20,150,269,178.

[9] Salvatore Sanfilippo. Redis new data structure: The HyperLogLog. `http://antirez.com/news/75`, 2014.

[10] Aiyou Chen, Jin Cao, Larry Shepp, and Tuan Nguyen. Distinct counting with a self-learning bitmap. *Journal of the American Statistical Association*, 106(495):879–890, 2011.

[11] Edith Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 88–99, New York, NY, USA, 2014.

[12] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182 – 209, 1985.

[13] Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15(2):208–229, 1990.

[14] Philippe Jacquet and Wojciech Szpankowski. Analytical depoissonization and its applications. *Theoretical Computer Science*, 201(1):1–62, 1998.

[15] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 605–617, Budapest, Hungary, September 2003.

[16] George Casella and Roger L. Berger. *Statistical inference*. Duxbury, Pacific Grove, CA, USA, 2nd edition, 2002.

[17] Peter Clifford and Ioana A. Cosma. A statistical analysis of probabilistic counting algorithms. *Scandinavian Journal of Statistics*, 39(1):1–14, 2012.

[18] Anirban Dasgupta, Kevin Lang, Lee Rhodes, and Justin Thaler. A framework for estimating stream expression cardinalities. *arXiv preprint arXiv:1510.01455*, 2015.

[19] Andrew Pascoe. Hyperloglog and MinHash - A union for intersections. `http://tech.adroll.com/media/hllminhash.pdf`, 2013.

[20] Reuven Cohen, Liran Katzir, and Aviv Yehezkel. A minimal variance estimator for the cardinality of big data set intersection. *arXiv preprint arXiv:1606.00996*, 2016.

[21] Daniel Ting. Towards optimal cardinality estimation of unions and intersections with sketches. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge and Data Mining*, pages 1195–1204, San Francisco, CA, USA, August 2016.

[22] William H. Press. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge University Press, 2007.

[23] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[24] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets.* Cambridge University Press, 2014.

[25] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[26] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, Montreal, Canada, May 2002.

[27] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29, Positano, Italy, June 1997.

[28] Ping Li and Arnd Christian König. Theory and applications of b-bit minwise hashing. *Communications of the ACM*, 54(8):101–109, 2011.

[29] Ping Li, Art Owen, and Cun hui Zhang. One permutation hashing. *Advances in Neural Information Processing Systems*, 25:3113–3121, 2012.