
# Efficient framework for operating on data sketches

Jakub Lemiesz
Wrocław University of Science and Technology
Wrocław, Poland
jakub.lemiesz@pwr.edu.pl

## ABSTRACT

We study the problem of analyzing massive data streams based on concise data sketches. Recently, a number of papers have investigated how to estimate the results of set-theory operations based on sketches. In this paper we present a framework that allows to estimate the result of any sequence of set-theory operations.

The starting point for our solution is the solution from 2021. Compared to this solution, the newly presented sketching algorithm is much more computationally efficient as it requires on average $O(\log n)$ rather than $O(n)$ comparisons for $n$ stream elements. We also show that the estimator dedicated to sketches proposed in that reference solution is, in fact, a maximum likelihood estimator.

**PVLDB Reference Format:**
Jakub Lemiesz. Efficient framework for operating on data sketches. PVLDB, 16(8): 1967 - 1978, 2023.
doi:10.14778/3594512.3594526

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://doi.org/10.6084/m9.figshare.22095122.v1

## 1 INTRODUCTION

### 1.1 Data sketches

While dealing with massive data streams, it is often infeasible in the long term to store all data for future analysis. Typically, a concise summary - in the literature often referred as data sketch - that allows to keep crucial information should be created online, as the elements of a stream are observed. The standard way to create a data sketch is based on using hash functions to assign to identical elements of the stream the same pseudorandom value.

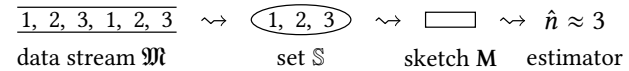



**Figure 1: The typical schema of creating a data sketch.**

The schema goes as follows (see: Figure 1). We model a stream as a multiset $\mathfrak{M} = (\mathbb{S}, m)$, where $\mathbb{S}$ is called the underlying set of elements and $m : \mathbb{S} \to \mathbb{N}_{\geq 1}$ is a function that determines the multiplicity of the elements. By using the hash function, we get rid of information about duplicates and create a sketch $\mathbf{M}$ that summarizes the underling set $\mathbb{S}$. Next, based on the sketch $\mathbf{M}$ we try to construct estimators to infer about some properties of set $\mathbb{S}$.

The best known solutions that fit into this scheme are cardinality estimation algorithms allowing to estimate the number $n$ of distinct stream elements by using storage that is sub-linear in $n$. In this category we have well-known algorithms such as KMV [4, 6] or HyperLogLog [12, 15, 16].

The generalized version of the cardinality estimation problem takes into account the weights of elements. Namely, the task is to estimate the total sum of weights of unique stream elements $|\mathbb{S}|_w = \sum_{i \in \mathbb{S}} \lambda_i$ , where an element $i$ has immutable weight $\lambda_i \in \mathbb{R}_+$. The weighted cardinality estimation problem was analyzed, for example, in [3, 7, 11, 18, 19]. Many practical applications for the weighted version of the problem are discussed in [18]. In particular, it is applicable in network traffic analysis, executing queries to distributed data sets or data aggregation in wireless sensor networks.

### 1.2 Operations on data sketches

The idea of using data sketches to estimate results of the set theory operations on corresponding data sets has been lately considered in a number of papers (see [2, 5, 8, 13, 18, 22]). For a thorough and up-to-date overview of the previous results see [18], which is a starting point of the solution presented in this paper. In essence, common to all these solutions is that it is quite straightforward to estimate the cardinality of the unions $|\mathbb{A}_1 \cup \mathbb{A}_2 \cup \ldots \mathbb{A}_k|$ based on sketches $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k$ that represent sets $\mathbb{A}_1, \mathbb{A}_2, \ldots \mathbb{A}_k$ . However, what is crucial, estimating an intersection $|\mathbb{A}_1 \cap \mathbb{A}_2 \cap \ldots \mathbb{A}_k|$ is significantly more challenging. Obviously, one could try using the inclusion-exclusion principle $|\mathbb{A}_1 \cap \mathbb{A}_2| = |\mathbb{A}_1| + |\mathbb{A}_2| - |\mathbb{A}_1 \cup \mathbb{A}_2|$ , but for larger number of sets it becomes very inaccurate and expensive (see [8]). Luckily, in some solutions it turns out that based on sketches $\mathbf{S}_1$ and $\mathbf{S}_2$ one could quite naturally and efficiently estimate Jaccard Similarity:

$$J(\mathbb{A}_1, \mathbb{A}_2) = \frac{|\mathbb{A}_1 \cap \mathbb{A}_2|}{|\mathbb{A}_1 \cup \mathbb{A}_2|} .$$

Then, by taking the product of estimates of $J(\mathbb{A}_1, \mathbb{A}_2)$ and $|\mathbb{A}_1 \cup \mathbb{A}_2|$ we can get a reasonable estimate of $|\mathbb{A}_1 \cap \mathbb{A}_2|$.

As shown in [18], this idea can be generalized to a larger number of sets and can also be used to estimate the weighted cardinality. The trick is to estimate the generalized weighted Jaccard Similarity:

$$J_w(\mathbb{A}_1, \mathbb{A}_2, \ldots \mathbb{A}_k) = \frac{|\mathbb{A}_1 \cap \mathbb{A}_2 \cap \ldots \mathbb{A}_k|_w}{|\mathbb{A}_1 \cup \mathbb{A}_2 \cup \ldots \mathbb{A}_k|_w} .$$

Based on this trick paper [18] propose a solution that allows to estimate the result of using the same operation multiple times. Namely, the proposed solution allows to estimate:

$$|\mathbb{A}_1 \cap \mathbb{A}_2 \cap \ldots \mathbb{A}_k|_w \quad \text{and} \quad |\mathbb{A}_1 \cup \mathbb{A}_2 \cup \ldots \mathbb{A}_k|_w .$$


Proceedings of the VLDB Endowment, Vol. 16, No. 8 ISSN 2150-8097.
doi:10.14778/3594512.3594526

### 1.3 Our contribution

There are three main contributions of this paper. First, we offer a generalization that allows to estimate the weighted cardinality of a set constructed from *any sequence* of set-theory operations.

Second, we significantly reduce the time of creating a sketch by reducing the average number of comparisons from $O(n)$ to $O(\log n)$, where $n$ is the number of stream elements.

Third, we show that the estimator proposed in [18] and dedicated to considered sketches is, in fact, a maximum likelihood estimator.

## 2 EXP-SKETCH ALGORITHM

In this section we briefly describe the algorithm proposed in [18] that is the starting point for further development (see: Algorithm 1). We assume that elements of the stream $\mathfrak{M}$ are pairs $(i, \lambda_i)$, where $i$ is a unique identifier and $\lambda_i \in \mathbb{R}_+$ is a weight. Without loss of generality we also assume that if there are $n$ distinct elements in a stream then $i \in \{1, 2, \ldots, n\}$. The problem is to construct a data sketch that allows to estimate $\Lambda = \lambda_1 + \lambda_2 + \ldots \lambda_n$ .

Note that in the algorithm presented in [18] each element has $d$ weights and each weight is connected to a different feature. However, each feature correspond to a separate row of a data sketch and could be considered separately. Therefore, for the sake of clarity, we consider only elements with one feature.

At first, for each element $(i, \lambda_i)$ one calculates $m$ hashes:

$$h(i \,||\, 1)\,,\ h(i \,||\, 2)\,,\ \ldots\ ,\ h(i \,||\, m)\,,$$

where symbol $i \,||\, k$ denotes the concatenation of two numbers $i$ and $k$, each represented by a fixed length binary sequence. For the purpose of the formal analysis hashes are considered to be independent random variables uniformly distributed in the unit interval. Each of $m$ generated hashes is transformed using the inverse cumulative distribution function of the exponential distribution with parameter $\lambda_i$ :

$$F^{-1}(u) = -\frac{\ln u}{\lambda_i}\ .$$

From the inverse transform sampling theorem (see [10]) follows that for each $k = 1, \ldots, m$ we obtain an exponentially distributed random variable:

$$-\frac{\ln h\,(i \,||\, k)}{\lambda_i} \sim Exp(\lambda_i)\ .$$

Then, in line 5 of Algorithm 1 we compare point-wise these $m$ random variables with the values currently stored in sketch $\mathbf{M}$.

**Algorithm 1** ExpSketch ( $\mathfrak{M}$ , $m$ )

**Initialization:**

1: set each of $m$ positions of sketch $\mathbf{M} = (\mathbf{M}_1, \ldots, \mathbf{M}_m)$ to $\infty$

**Upon arrival of an element** $(i, \lambda_i) \in \mathfrak{M}$ **:**

2: **for all** $k \in \{1, 2, \ldots, m\}$ **do**
3: $\quad U \leftarrow h\,(i \,||\, k)$
4: $\quad E \leftarrow -\ln U / \lambda_i$
5: $\quad \mathbf{M}_k \leftarrow \min\{\mathbf{M}_k\,,\, E\}$
6: **end for**

**Upon request, at any time:**

7: **return:** $\mathbf{M}$

For $i = 1, \ldots, n$ let us define $E_i \sim Exp(\lambda_i)$ and note that a value stored in $\mathbf{M}_k$ after $n$ elements has been observed corresponds to

$$\mathbf{M}_k = \min\{E_1, E_2, \ldots, E_n\}\ . \tag{1}$$

Since the minimum of exponentially distributed random variables is also exponentially distributed we obtain that

$$\mathbf{M}_k \sim Exp(\Lambda)\ . \tag{2}$$

Note that random variables $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m$ are independent copies of the same experiment. The independence can be justified by the fact that all the hashes $h(i \,||\, k)$ generated for different values $i$ or $k$ are considered to be independent random variables.

The sum $G_m = \mathbf{M}_1 + \mathbf{M}_2 + \ldots + \mathbf{M}_m$ of $m$ independent exponentially distributed random variables with the same mean $\Lambda$ follows the gamma distribution: $G_m \sim \Gamma(m, \Lambda)$, where $m \in \mathbb{N}_+$ and $\Lambda \in \mathbb{R}_+$ . Based on the properties of gamma distribution, one can show that an estimator defined as

$$\overline{\Lambda} = \frac{m-1}{G_m} \tag{3}$$

for $m \geq 2$ is an unbiased estimator of the parameter $\Lambda$ , namely:

$$\mathbb{E}[\,\overline{\Lambda}\,] = \Lambda\ . \tag{4}$$

For $m \geq 3$ we get its variance and standard error:

$$\mathbb{V}\mathrm{ar}[\,\overline{\Lambda}\,] = \frac{\Lambda^2}{m-2} \quad \text{and} \quad \mathbb{SE}[\,\overline{\Lambda}\,] = 1\Big/\sqrt{m-2}\ . \tag{5}$$

The memory/precision trade-off is similar as for well known KMV sketches [6], which do not allow weights and offer only basic operations on sketches. In both solutions for a sketch with $m$ hash values we get $\mathbb{SE} \approx 1/\sqrt{m}$ . For the discussion on how to adjust the length of hash values to the number of distinct elements $n$ see [18].

## 3 MAXIMUM LIKELIHOOD ESTIMATOR

In [18] experimental results gave reason to believe that estimator $\overline{\Lambda}$ defined in formula (3) follows the normal distribution. In this section we present the proof that $\overline{\Lambda}$ is a maximum likelihood estimator. This guarantees its converges in distribution to a normal distribution as parameter $m$ increases. Moreover, it guarantees an asymptotic consistency and efficiency of estimator $\overline{\Lambda}$ with the growth of parameter $m$. Note that an efficient estimator is the minimum variance unbiased estimator. Let us also remark that in [3] the authors have proven the cardinality estimator analogous to estimator $\overline{\Lambda}$ (but not allowing weights) is a maximum likelihood estimator.

Consider random variables $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m$ corresponding to the content of the sketch in Algorithm 1. Since $\mathbf{M}_i \sim Exp(\Lambda)$ the probability density function of $\mathbf{M}_i$ is $f(x_i) = \Lambda e^{-\Lambda x_i}$ . Then, we can express the log-likelihood function in terms of unknown parameter $\Lambda$

$$\log L(\Lambda) \ = \ \log \prod_{i=1}^{m} \Lambda e^{-\Lambda x_i} \ = \ m \log(\Lambda) - \Lambda \sum_{i=1}^{m} x_i\ .$$

To find the value of parameter $\Lambda$ that maximizes the value of $\log L(\Lambda)$ we compare its first derivative to zero

$$\frac{\log L(\Lambda)}{d\Lambda} \ = \ \frac{m}{\Lambda} - \sum_{i}^{m} x_i \ = \ 0 \quad \text{and get} \quad \Lambda = \frac{m}{\sum_i^m x_i}\ .$$

Finally, by changing in the above formula $m$ to $m-1$ for bias correction and replacing $x_i$ by $\mathbf{M}_i$ we get estimator (3).

## 4 FAST-EXP-SKETCH ALGORITHM

All the computational cost in Algorithm 1 comes from generating for each element of the stream $m$ values and comparing them pointwise to the current content of the sketch $\mathbf{M}$. While the algorithm is very simple, in practice for large values of parameter $m$ and large number of stream elements the process of creating a sketch can take an unacceptable amount of time. From our experience, this is the case not only for devices with limited computational power like sensors, but also for more powerful machines (see Section 6.2).

In this section we will show how to significantly reduce the total number of values generated in Algorithm 1 and thus the number of required comparisons. The main idea is based on the following result (cf. [10]).

THEOREM 4.1. *Let $E_1, E_2, \ldots, E_m$ be a sequence of i.i.d. exponential random variables and let*

$$E_{(1)} \leq E_{(2)} \leq \ldots \leq E_{(m)}$$

*denote their order statistics, i.e. $E_{(k)}$ denotes the $k$-th smallest value. Then for each $k \in \{1, 2, \ldots, m\}$ we have the distributional equality:*

$$E_{(k)} \stackrel{\mathrm{d}}{=} E_{(k-1)} + \frac{E_k}{m-k+1} \,. \tag{6}$$

### 4.1 Algorithm construction

Note that in Algorithm 1 for each element $(i, \lambda_i)$ we generate $m$ values $E_1, E_2, \ldots E_m$ and $E_k \sim Exp(\lambda_i)$ for each $k \in \{1, 2, \ldots, m\}$. Then we compare these values with the values currently stored in the sketch and assign $\mathbf{M}_k \leftarrow \min\{\mathbf{M}_k, E_k\}$. Note that $E_k$ surely will not be assigned to $\mathbf{M}_k$ if

$$E_k > \max\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m\} \,.$$

Based on the above observation, we can get identical results as in the procedure described in Algorithm 1 but with much smaller number of generated values. Namely, rather than generating variables $E_1, E_2, \ldots$ as in Algorithm 1, we generate order statistics $E_{(1)}, E_{(2)}, \ldots$ in increasing order until we get $E_{(i)} > MAX$ and compare $E_{(1)}, E_{(2)}, \ldots, E_{(i-1)}$ to values at random positions of sketch $\mathbf{M}$. Let us remark that we cannot just compare $E_{(k)}$ to value of $\mathbf{M}_k$. In such an approach rather than obtaining i.i.d. random variables $\mathbf{M}_1, \ldots, \mathbf{M}_m$ we would have a positive correlation between the position of the sketch $k$ and the value stored in this position.

The pseudo-code precisely describing the new procedure is presented below as Algorithm 2. In lines 1-3 we initialize the variables. Upon arrival of a new element $(i, \lambda_i)$ in line 4 we reset to zero variable $S$ used to store the value of consecutive order statistics. In line 5 we reset variable *updateMAX* that indicates whether an element has caused the change of the maximal value in the sketch. In line 6 we reset variable $P$ used to store a random permutation generated for an element.

In line 7 we start the loop. In the $k$-th step of the loop we compute the $k$-th order statistic and assign it to a random position $j$ of $\mathbf{M}$ based on the random permutation $P$. Lines 8-9 generate pseudo-random values from the exponential distribution with parameter $\lambda_i$ and are exactly the same as in Algorithm 1. In line 10 we compute the value of the $k$-th order statistic based on the formula (6).

In line 11, if the value of the $k$-th order statistic is greater than the current value of $MAX$ then neither this order statistic, nor any subsequent will be stored in the sketch, thus we break the loop. Otherwise, in lines 12-13 we take a single step of Fisher–Yates shuffle to compute the value at the $k$-th position of permutation $P$. Note that in line 12 the seed is set to be an identifier $i$ and thus, if an element $(i, \lambda_i)$ appears in the stream multiple times, we always get the same associated permutation. Note also that after $k$-th step of Fisher–Yates shuffle the first $k$ positions of the permutation are computed and fixed (see e.g. [17] and Algorithm 3 below). Generating the whole permutation $P$ at once would be unnecessary and costly, because, as we will show, most often only a small initial part of $P$ is used before the loop is broken in line 11.

In line 14 we assign a random position $j$ for the $k$-th order statistic based on permutation $P$. In line 15 we check whether there is a maximum at position $j$ and in such case we set the flag *updateMax*, which indicates that the maximum has been changed. In line 16 we compare the $k$-th order statistic $S$ to the value stored in $\mathbf{M}_j$.

When the loop is finished and the maximum in the sketch has been changed, in line 18 we compute the new maximum.

**Algorithm 2** FastExpSketch ( $\mathfrak{M}$ , $m$ )

**Initialization:**

1: $permInit \leftarrow (1, 2, 3, \ldots, m)$ — $O(m)$
2: set all values of $\mathbf{M} = (\mathbf{M}_1, \ldots, \mathbf{M}_m)$ to $\infty$ — $O(m)$
3: $MAX \leftarrow \infty$ — $O(1)$

**Upon arrival of an element** $(i, \lambda_i) \in \mathfrak{M}$ **:**

4: $S \leftarrow 0$ — $O(1)$
5: $updateMAX \leftarrow false$ — $O(1)$
6: $P \leftarrow permInit$ — $O(1)$
7: **for all** $k \in \{1, 2, \ldots, m\}$ **do**
8: $\quad U \leftarrow h(i \,||\, k)$ — $O(1)$
9: $\quad E \leftarrow -\ln U / \lambda_i$ — $O(1)$
10: $\quad S \leftarrow S + E/(m-k+1)$ — $O(1)$
11: $\quad$ **if** $S > MAX$ **then** break — $O(1)$
12: $\quad r \leftarrow RandomInteger([k, m], seed = i)$ — $O(1)$
13: $\quad$ exchange $P[k]$ and $P[r]$ — $O(1)$
14: $\quad j \leftarrow P[k]$ — $O(1)$
15: $\quad$ **if** $\mathbf{M}_j == MAX$ **then** $updateMAX \leftarrow true$ — $O(1)$
16: $\quad \mathbf{M}_j \leftarrow \min\{\mathbf{M}_j, S\}$ — $O(1)$
17: **end for**
18: **if** updateMAX **then** $MAX \leftarrow \max\{\mathbf{M}_1, \ldots, \mathbf{M}_m\}$ — $O(m)$

**Upon request, at any time:**

19: **return:** $\mathbf{M}$

**Algorithm 3** Fisher–Yates shuffle

1: $P \leftarrow (1, 2, 3, \ldots, m)$
2: **for all** $k \in \{1, 2, \ldots, m\}$ **do**
3: $\quad r \leftarrow RandomInteger([k, m])$
4: $\quad$ exchange $P[k]$ and $P[r]$
5: **end for**
6: **return:** $P$

## 4.2 Average number of comparisons

In Algorithm 1 for each element of a stream, we have always $m$ executions of the loop in line 2 or equivalently $m$ comparisons in line 5. Thus, in total for $n$ elements there are $m \cdot n$ comparisons. In this section we will show that in case of Algorithm 2 for most cases we have on the average $O(\log n)$ comparisons in total.

Without loss of generality we assume $(i, \lambda_i)$ is the $i$-th element of a stream (note $i$ is only used as an input to the hash function). Let $MAX_i$ denotes the value of the maximum used in line 11 and let $C_i$ be a random variable denoting the number of comparisons in line 16 of Algorithm 2 for this $i$-th element. Note that for element $(1, \lambda_1)$ we have $MAX_1 = \infty$ and the condition in line 11 never holds. Thus, we have $C_1 = m$.

Theorem 4.2. *For $i \geq 2$,*

$$\Lambda_i = \lambda_1 + \lambda_2 + \ldots \lambda_i \quad \text{and} \quad r_i = \frac{\lambda_i}{\Lambda_{i-1}}$$

*we have*

$$\mathbb{E}[C_i] = m\left(1 - \prod_{j=1}^{m} \frac{j}{j + r_i}\right). \tag{7}$$

Proof. Since always $0 \leq C_i \leq m$ we get:

$$\mathbb{E}[C_i] = \sum_{k=1}^{m} k \Pr[C_i = k] = \sum_{k=1}^{m} \Pr[C_i \geq k] \ . \tag{8}$$

Note that

$$\Pr[C_i \geq k] = \Pr\left[E_{(k)} < MAX_i\right] \ , \tag{9}$$

where $E_{(k)}$ corresponds to the $k$-th order statistic computed in line 10 of Algorithm 2 (cf. Section 4.1). Note also that we have

$$\begin{aligned}
&\Pr\left[E_{(k)} \leq MAX_i\right] = \\
&\int_0^\infty \Pr\left[E_{(k)} \leq MAX_i \mid MAX_i = x\right] \ m_i(x)\ dx = \\
&\int_0^\infty \Pr\left[E_{(k)} \leq x\right] \ m_i(x)\ dx = \\
&\int_0^\infty F_k(x)\ m_i(x)\ dx \,,
\end{aligned} \tag{10}$$

where

$$m_i(x) = e^{-\Lambda_{i-1}x}\left(1 - e^{-\Lambda_{i-1}x}\right)^{m-1} m\,\Lambda_{i-1} \tag{11}$$

is the probability density function of the random variable $MAX_i$ (see Lemma 4.3) and

$$F_k(x) = \sum_{j=k}^{m} \binom{m}{j} p^j (1-p)^{m-j} \quad \text{for} \quad p = 1 - e^{-\lambda_i x} \tag{12}$$

is the cumulative density function of the $k$-th order statistic $E_{(k)}$ for random variables $E_1, E_2, \ldots, E_m \sim Exp(\lambda_i)$ (see Lemma 4.4).

By using formulas (8), (9) and (10) we get

$$\mathbb{E}[C_i] = \int_0^\infty \left(\sum_{k=1}^{m} F_k(x)\right) \ m_i(x)\ dx \,, \tag{13}$$

where by the Lemma 4.5 we have $\sum_{k=1}^{m} F_k(x) = m\left(1 - e^{-\lambda_i x}\right)$ .

Next, by using standard techniques and the properties of the gamma function $\Gamma(x)$, we can calculate the value of the definite integral in the formula (13):

$$\mathbb{E}[C_i] = m\left(1 - \frac{\Gamma(m+1)\Gamma(1+r_i)}{\Gamma(m+1+r_i)}\right).$$

The above result of the integration can also be easily verified in a symbolic computation system (e.g. Wolfram Mathematica).

Finally, to obtain the formula (7), we use the properties of the gamma function (see e.g. [1]). Namely, we use the fact that for $m \in \mathbb{N}$ and $x \in \mathbb{R}_+$ we have $\Gamma(m+1) = m!$ and

$$\frac{\Gamma(1+x)}{\Gamma(m+1+x)} = \frac{1}{\prod_{j=1}^{m}(j+x)} \ .$$

□

Lemma 4.3. *Let $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m$ be independent random variables such that for each $k$ we have $\mathbf{M}_k \sim Exp(\Lambda_{i-1})$ and*

$$MAX_i = \max\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m\} \ .$$

*Then the probability density function $m_i(x)$ of random variable $MAX_i$ is given by the formula (11).*

Proof. Since $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m$ are independent, we can easily obtain the cumulative distribution function of $MAX_i$ :

$$\begin{aligned}
\Pr[MAX_i \leq x] &= \Pr[\mathbf{M}_1 \leq x \ \wedge \ \ldots \ \wedge \mathbf{M}_m \leq x] \\
&= \Pr[M_1 \leq x]^m = \left(1 - e^{-\Lambda_{i-1}x}\right)^m \ .
\end{aligned}$$

To obtain formula (11) one can just compute derivative of the above function in terms of variable $x$. □

Lemma 4.4. *Let $E_1, \ldots, E_m \sim Exp(\lambda_i)$ be i.i.d. random variables. The cumulative distribution function of their $k$-th order statistic $E_{(k)}$ is given by the formula:*

$$F_k(x) = \sum_{j=k}^{m} \binom{m}{j} p^j (1-p)^{m-j} \quad \text{where} \quad p = 1 - e^{-\lambda_i x} \ .$$

Proof. Note for each $j \in \{1, 2 \ldots, m\}$ we have $p = \Pr\left[E_j \leq x\right]$. Observe that $E_{(k)} \leq x$ if and only if at least $k$ of variables $E_1, \ldots, E_m$ is not greater then $x$. Therefore, we can define a random variable with the binomial distribution $X_{m,p} \sim Bin(m, p)$ and note that

$$F_k(x) = \Pr\left[E_{(k)} \leq x\right] = \Pr\left[X_{m,p} \geq k\right] \ .$$

□

Lemma 4.5. *Let $F_k(x)$ be defined as in Lemma 4.4. Then we have*

$$\sum_{k=1}^{m} F_k(x) = m\left(1 - e^{-\lambda_i x}\right) \ .$$

Proof.

$$\begin{aligned}
\sum_{k=1}^{m} F_k(x) &= \sum_{k=1}^{m} \sum_{j=k}^{m} \binom{m}{j} p^j (1-p)^{m-j} \\
&= \sum_{j=1}^{m} j \binom{m}{j} p^j (1-p)^{m-j} = mp \ .
\end{aligned}$$

The last equality follows from the fact that $\mathbb{E}\left[X_{m,p}\right] = mp$. □

### 4.3 The edge case scenarios

The general conclusion from Theorem 4.2 is that the total average number of comparisons $\mathbb{E}[C] = \mathbb{E}[C_1] + \ldots + \mathbb{E}[C_n]$ depends on the value of the ratio

$$r_i = \frac{\lambda_i}{\Lambda_{i-1}} \quad \text{where} \quad \Lambda_i = \lambda_1 + \lambda_2 + \ldots + \lambda_i$$

for consecutive elements $i \in \{1, \ldots, n\}$. In this section we analyze the edge case scenarios.

**Scenario 1.** Assume $\lambda_1 = \lambda_2 = \ldots = \lambda_n$. From Theorem 4.2 for $i \geq 2$ we get

$$r_i = \frac{1}{i-1} \qquad \text{and} \qquad \mathbb{E}[C_i] = m\left(1 - \prod_{j=1}^{m} \frac{j}{j + \frac{1}{i-1}}\right) .$$

Using properties of Stirling numbers of the first kind (or Wolfram Mathematica) it is easy to show that in this case

$$\lim_{i\to\infty} i\,\mathbb{E}[C_i] = m\,H_m \qquad \text{and} \qquad \mathbb{E}[C_i] \leq \frac{m\,H_m}{i} ,$$

where $H_m$ is the $m$-th harmonic number. Therefore,

$$\mathbb{E}[C] \leq \sum_{i=1}^{n} \frac{m\,H_m}{i} = m\,H_m\,H_n ,$$

where $H_n = \ln n + \gamma + O(1/n)$ , while in the original algorithm we would have $\mathbb{E}[C] = m \cdot n$. Note that $m$ is a sketch size and is considered to be constant.

From Scenario 1 we can deduce that if values of weights $\lambda_1, \ldots, \lambda_n$ are comparable then $r_i \approx \frac{1}{i}$ and $\mathbb{E}[C]$ is logarithmic in $n$. In the next example we consider scenario in which weights of consecutive elements systematically grows.

**Scenario 2.** Assume $\lambda_i = i$ for each $i$. Then for $i \geq 2$

$$\Lambda_{i-1} = \frac{(i-1)i}{2} \qquad \text{and} \qquad r_i = \frac{2}{i-1} .$$

By using exactly the same line of reasoning as in the Scenario 1 we can show that in this case

$$\mathbb{E}[C] \leq \sum_{i=1}^{n} \frac{2\,m\,H_m}{i} = 2\,m\,H_m\,H_n .$$

Note that if in the Scenario 2 we scale all elements by some factor $f > 1$ to increase the distance between them, we would get exactly the same ratio $r_i = \frac{2}{i-1}$ and thus the same bound on $\mathbb{E}[C]$.

Finally, we consider an extremely pessimistic scenario, in which the weights of consecutive elements increase exponentially.

**Scenario 3.** Assume $\lambda_1 = 1$ and for $i \geq 2$ we have $\lambda_i = 2^{i-2}$. Then for $i \geq 2$

$$\Lambda_{i-1} = 2^{i-2} \qquad \text{and} \qquad r_i = 1 .$$

Thus we have

$$\mathbb{E}[C] = m + \sum_{k=2}^{n} m\left(1 - \prod_{j=1}^{m} \frac{j}{j+1}\right) = m + \sum_{k=2}^{n} m\left(1 - \frac{1}{m+1}\right)$$
$$= m\left(1 + (n-1)\left(1 - \frac{1}{m+1}\right)\right) \approx m \cdot n .$$

Note that Scenario 3 is the worst-case scenario in the sense that for each element we almost always have the maximal possible number of comparisons, which is $m$, and thus $m \cdot n$ comparisons in total.

### 4.4 Average-case time complexity

Let us note that in the previous section we consider the pessimistic scenarios in the sense that elements appear in the ascending order. In the average case the order of the elements could be expected to be more random, and thus the values of ratio $r_i$ for consecutive elements should be smaller, which in turn reduce values of $\mathbb{E}[C_i]$.

In general, if we assume that for element $(i, \lambda_i)$ we have

$$r_i = \frac{\lambda_i}{\Lambda_{i-1}} = \frac{1}{m^{\alpha+1}} \qquad \text{for} \qquad \alpha \geq 0$$

we can show that

$$\mathbb{E}[C_i] = \frac{\ln m + \gamma}{m^{\alpha}} + O\left(\frac{1}{m}\right) .$$

This fact can be interpreted as follows. For some number of initial elements, for which the value of $r_i$ is large, the number of comparisons is also large. However, as soon as the ratio $r_i$ drops below $1/m^2$, the value of $\mathbb{E}[C_i]$ starts approaching zero. Therefore, even if the number of elements $n$ grows, we can still have $\mathbb{E}[C] = O(\ln n)$.

Note that the above analysis concerns the number of comparisons and does not include the pre-processing of stream elements. Let $t_2$ be the time to pre-process a stream element in Algorithm 2 (element pre-processing = all steps up to the first call of line 11). Note also that there is no pre-processing of elements in Algorithm 1 and let $t_1$ be the time of a single loop execution in this algorithm. Therefore, if we include pre-processing, the total time for $n$ elements in Algorithm 1 is $t_1 mn$ and in Algorithm 2 is $t_2 n + O(\ln n)$. Term $O(\ln n)$ follows from the fact that the total number of loop steps in line 7 corresponds to the total number of comparisons $C$. Moreover, computing the maximum in line 18 can be executed only for elements for which we have at least one comparison. Note that the number of such elements can not be greater than the total number of comparisons $C$. Obviously, for sufficiently large $n$ term $t_2 n$ dominates $O(\ln n)$ and then Algorithm 2 is approximately $m$ times faster than Algorithm 1 (since times $t_1$ and $t_2$ are similar).

In the analysis presented in this section we do not consider the possibility of multiple occurrences of an element in a stream. Note that only the first occurrence of an element affects the sum $\Lambda_i$. Note also that for each occurrence of an element $(i, \lambda_i)$ the number of comparisons in line 16 of Algorithm 2 depends on the ratio of $\lambda_i$ and the current sum of weights of different elements that appeared before the considered instance of $(i, \lambda_i)$. Therefore, repetitions of elements do not significantly affect the results presented in this section, except in situations where there is a small number of distinct elements in the stream and the ratio $r_i$ does not decrease with $i$ (see experiments in Section 6.1, in particular Figure 3c).

## 5 SET-THEORY OPERATIONS ON SKETCHES

In this section we show how the results presented in [18] can be generalized to allow estimating the weighted cardinality of a set constructed from *any sequence* of set-theory operations.

### 5.1 Summary of previous results

First, let us briefly summarize the results obtained in [18]. Assume we have two sketches created by Algorithm 1 for sets $\mathbb{A}$ and $\mathbb{B}$:

$$\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m) \qquad \text{and} \qquad \mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_m) .$$

Based on the formulas (1) and (2) we know that $\mathbf{A}_k$ and $\mathbf{B}_k$ are minimums of generated values and therefore correspond to exponentially distributed random variables:

$$\mathbf{A}_k \sim Exp(\,|\mathbb{A}|_w\,) \quad \text{and} \quad \mathbf{B}_k \sim Exp(\,|\mathbb{B}|_w\,)\,.$$

*5.1.1 Sum.* To estimate $|\mathbb{A}\cup\mathbb{B}|_w$ based on sketches $\mathbf{A}$ and $\mathbf{B}$ we can create sketch $\mathbf{A}\cup\mathbf{B}$ by using the point-wise minimum:

$$\mathbf{A}\cup\mathbf{B} := (\,\min\{\mathbf{A}_1,\mathbf{B}_1\}\,,\,\ldots\,,\,\min\{\mathbf{A}_m,\mathbf{B}_m\}\,)\,.$$

Namely, since each position of the sketch is a minimum and since

$$\min\{\,\min\{\mathbb{A}\}\,,\,\min\{\mathbb{B}\}\,\} = \min\{\mathbb{A}\cup\mathbb{B}\}\,,$$

then sketch $\mathbf{A}\cup\mathbf{B}$ corresponds to the sketch we would get by observing elements of set $\mathbb{A}\cup\mathbb{B}$ and for its positions we have:

$$\min\{\mathbf{A}_k,\mathbf{B}_k\} \sim Exp(\,|\mathbb{A}\cup\mathbb{B}|_w\,)\,.$$

Therefore, based on definition (3) we can define an unbiased estimator of $|\mathbb{A}\cup\mathbb{B}|_w$ for $m\geq 2$ as

$$\overline{U}(\mathbf{A},\mathbf{B}) := \frac{m-1}{\sum_{k=1}^{m}\min\{\mathbf{A}_k,\mathbf{B}_k\}}\,. \tag{14}$$

This approach can be generalized for any number of sketches $\overline{U}(\mathbf{A},\mathbf{B},\mathbf{C},\ldots)$ by changing summands in (14) to $\min\{\mathbf{A}_k,\mathbf{B}_k,\mathbf{C}_k,\ldots\}$.

*5.1.2 Jaccard Similarity.* The weighted Jaccard similarity on sets $\mathbb{A}$ and $\mathbb{B}$ can be defined as

$$J_w(\mathbb{A},\mathbb{B}) := \frac{|\mathbb{A}\cap\mathbb{B}|_w}{|\mathbb{A}\cup\mathbb{B}|_w}. \tag{15}$$

In [18] it has been shown that

$$J_w(\mathbb{A},\mathbb{B}) = \Pr[\mathbf{A}_k=\mathbf{B}_k] \tag{16}$$

and thus for an estimator based on the Iverson bracket notation

$$\bar{J}_w(\mathbf{A},\mathbf{B}) := \frac{1}{m}\sum_{k=1}^{m}[\![\mathbf{A}_k=\mathbf{B}_k]\!] \tag{17}$$

we have

$$\mathbb{E}[\bar{J}_w(\mathbf{A},\mathbf{B})] = J_w(\mathbb{A},\mathbb{B})$$

and

$$\mathbb{V}\mathrm{ar}\left[\bar{J}_w(\mathbf{A},\mathbf{B})\right] = \frac{J_w(\mathbb{A},\mathbb{B})\,(1-J_w(\mathbb{A},\mathbb{B}))}{m}\,.$$

Formula (15) can be generalized to a larger number of sets

$$J_w(\mathbb{A},\mathbb{B},\mathbb{C},\ldots) := \frac{|\mathbb{A}\cap\mathbb{B}\cap\mathbb{C}\ldots|_w}{|\mathbb{A}\cup\mathbb{B}\cup\mathbb{C}\ldots|_w} \tag{18}$$

and it has been shown that estimator

$$\bar{J}_w(\mathbf{A},\mathbf{B},\mathbf{C},\ldots) := \frac{1}{m}\sum_{k=1}^{m}[\![\mathbf{A}_k=\mathbf{B}_k=\mathbf{C}_k=\ldots]\!] \tag{19}$$

is unbiased estimator of (18).

*5.1.3 Intersection of sketches.* Using sketches $\mathbf{A}$ and $\mathbf{B}$ one can define an estimator of the intersection $|\mathbb{A}\cap\mathbb{B}|_w$ as:

$$\bar{I}(\mathbf{A},\mathbf{B}) := \bar{J}_w(\mathbf{A},\mathbf{B})\;\overline{U}(\mathbf{A},\mathbf{B})\,. \tag{20}$$

In [18] it has been shown that for $m\geq 2$ we have

$$\mathbb{E}\left[\bar{I}(\mathbf{A},\mathbf{B})\right] = |\mathbb{A}\cap\mathbb{B}|_w$$

and for $m\geq 3$, $p=|\mathbb{A}\cap\mathbb{B}|_w$ and $s=|\mathbb{A}\cup\mathbb{B}|_w$ we get:

$$\mathbb{V}\mathrm{ar}\left[\bar{I}(\mathbf{A},\mathbf{B})\right] = \frac{p^2}{(m-2)m} + \frac{(m-1)ps}{(m-2)m} \approx \frac{ps}{m}\,. \tag{21}$$

Moreover, the estimator

$$\bar{I}(\mathbf{A},\mathbf{B},\mathbf{C},\ldots) := \bar{J}_w(\mathbf{A},\mathbf{B},\mathbf{C},\ldots)\,\overline{U}(\mathbf{A},\mathbf{B},\mathbf{C},\ldots) \tag{22}$$

is an unbiased estimator of $|\mathbb{A}\cap\mathbb{B}\cap\mathbb{C}\ldots|_w$ and its variance can be expressed as in equation (21) with $p=|\mathbb{A}\cap\mathbb{B}\cap\mathbb{C}\ldots|_w$ and $s=|\mathbb{A}\cup\mathbb{B}\cup\mathbb{C}\ldots|_w$.

Let us emphasize that formula (22) is in practice very easy to use. Namely, to estimate $J_w$ we use formula (19) and find a number $m'$ of positions that are equal in all the sketches. Next, to estimate the weighted cardinality of the sum of all sets, we use formula (14) with the minimum $m_k=\min\{\mathbf{A}_k,\mathbf{B}_k,\mathbf{C}_k,\ldots\}$ for each position $k$. Finally, we deal with a neat formula:

$$\bar{I}(\mathbf{A},\mathbf{B},\mathbf{C},\ldots) = \frac{m'}{m}\,\frac{m-1}{\sum_{k=1}^{m}m_k}\,. \tag{23}$$

*5.1.4 Relative complement.* Similarly as in formula (16) for the weighted Jaccrad similarity one can show that for the complement over union we have

$$\frac{|\mathbb{A}\setminus\mathbb{B}|_w}{|\mathbb{A}\cup\mathbb{B}|_w} = \Pr[\mathbf{A}_k<\mathbf{B}_k]\,.$$

Thus, by analogy to the weighted Jaccard similarity estimator (17) an unbiased estimator of the complement over union has been defined in [18] as

$$\overline{CoU}(\mathbf{A},\mathbf{B}) := \frac{1}{m}\sum_{k=1}^{m}[\![\mathbf{A}_k<\mathbf{B}_k]\!]\,. \tag{24}$$

Then, it has been shown that for $m\geq 2$ and estimator

$$\bar{R}(\mathbf{A},\mathbf{B}) := \overline{CoU}(\mathbf{A},\mathbf{B})\,\overline{U}(\mathbf{A},\mathbf{B}) \tag{25}$$

we have

$$\mathbb{E}[\bar{R}(\mathbf{A},\mathbf{B})] = |\mathbb{A}\setminus\mathbb{B}|_w\,.$$

The variance of estimator $\bar{R}(\mathbf{A},\mathbf{B})$ can be expressed by formula (21) with parameter $p$ redefined to $p=|\mathbb{A}\setminus\mathbb{B}|_w$.

The value of estimator (25) is also easy to evaluate in practice. Namely, one can use formula (23) with

$$m' = \sum_{k=1}^{m}[\![\mathbf{A}_k<\mathbf{B}_k]\!] \quad \text{and} \quad m_k=\min\{\mathbf{A}_k,\mathbf{B}_k\}\,.$$

## 5.2 Scheme for three sets

In this section we will show how to generalize the solution described in the previous section in such a way that any sequence of set-theory operations on three sets can be mimic with the corresponding operations on sketches. In the next section we will show that the proposed scheme can be generalized also for a larger number of sets.

First, let us note that any expression consisted of set-theory operations can be transformed to a full disjunctive normal form. Namely, it can be expressed as the sum of disjoint intersections of sets and in every intersection each set appears exactly once. The appearance could be a set itself or its complement (see Figure 2).

For example, let us consider three sets $\mathbb{A}, \mathbb{B}, \mathbb{C}$ and the following sequence of operations, where for the sake of brevity of the notation, we omit the intersection symbol:

$$(\mathbb{A} \setminus \mathbb{C}) \cup \mathbb{A}\mathbb{B}\mathbb{C} = \mathbb{A}\mathbb{B}\mathbb{C} \cup \mathbb{A}\mathbb{B}\overline{\mathbb{C}} \cup \mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}\,. \tag{26}$$

By $\overline{\mathbb{A}} = \Omega \setminus \mathbb{A}$ we denote the set complement, and we assume that

$$\Omega = \mathbb{A} \cup \mathbb{B} \cup \mathbb{C}\,.$$

Therefore, we know that $\overline{\mathbb{A}}\,\overline{\mathbb{B}}\,\overline{\mathbb{C}} = \emptyset$ and thus $|\overline{\mathbb{A}}\,\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w = 0$.

The expression on the right hand side of equation (26) is in a full disjunctive normal form. Since all intersections in this expression are disjoint, to estimate

$$|(\mathbb{A} \setminus \mathbb{C}) \cup \mathbb{A}\mathbb{B}\mathbb{C}\,|_w$$

one can estimate the weighted cardinality of each intersection separately:

$$|\mathbb{A}\mathbb{B}\mathbb{C}|_w \quad , \quad |\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w \quad , \quad |\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w\,.$$

Obviously, to estimate $|\mathbb{A}\mathbb{B}\mathbb{C}|_w$ we can use formula (23). To estimate $|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w$ let us prove the following lemma, which is an extension of Lemma 4.1 from [18].

Lemma 5.1.

$$\Pr[\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k] = \frac{|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w}{|\Omega|_w}\,. \tag{27}$$

Proof. For sets $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ let $\mathbb{S}_\mathbb{A}$, $\mathbb{S}_\mathbb{B}$, $\mathbb{S}_\mathbb{C}$ be sets of values generated in line 4 of Algorithm 1 for a fixed position $k$ of a sketch. Note that

$$\mathbf{A}_k = \min\{\mathbb{S}_\mathbb{A}\} \sim Exp\,(\,|\mathbb{A}|_w\,)\,,$$
$$\mathbf{B}_k = \min\{\mathbb{S}_\mathbb{B}\} \sim Exp\,(\,|\mathbb{B}|_w\,)\,,$$
$$\mathbf{C}_k = \min\{\mathbb{S}_\mathbb{C}\} \sim Exp\,(\,|\mathbb{C}|_w\,)\,.$$

Consider element $(i, \lambda_i)$ and corresponding value $E_i$ generated in line 4 of Algorithm 1. Surely, if $(i, \lambda_i) \in \mathbb{A}\mathbb{B}\overline{\mathbb{C}}$ then $E_i \in \mathbb{S}_\mathbb{A}\,\mathbb{S}_\mathbb{B}\,\overline{\mathbb{S}}_\mathbb{C}$. The reverse implication also holds except the negligible collisions. Then we must have

$$\Pr[\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k] = \Pr\left[\min\left\{\mathbb{S}_\mathbb{A}\,\mathbb{S}_\mathbb{B}\,\overline{\mathbb{S}}_\mathbb{C}\right\} < \min\left\{\Omega_\mathbb{S} \setminus \mathbb{S}_\mathbb{A}\,\mathbb{S}_\mathbb{B}\,\overline{\mathbb{S}}_\mathbb{C}\right\}\right]\,,$$

where $\Omega_\mathbb{S} = \mathbb{S}_\mathbb{A} \cup \mathbb{S}_\mathbb{B} \cup \mathbb{S}_\mathbb{C}$.

Finally, since the minimum of exponentially distributed random variables is also exponentially distributed:

$$\min\left\{\mathbb{S}_\mathbb{A}\,\mathbb{S}_\mathbb{B}\,\overline{\mathbb{S}}_\mathbb{C}\right\} \sim Exp\left(\,|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w\,\right)\,,$$
$$\min\left\{\Omega_\mathbb{S} \setminus \mathbb{S}_\mathbb{A}\,\mathbb{S}_\mathbb{B}\,\overline{\mathbb{S}}_\mathbb{C}\right\} \sim Exp\left(\,|\Omega \setminus \mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w\,\right)$$

and since for independent

$$X \sim Exp(x) \quad \text{and} \quad Y \sim Exp(y)$$

we have

$$\Pr[X < Y] = x/(x+y)$$

we can write

$$\Pr[\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k] = \frac{|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w}{|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w + |\Omega \setminus \mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w}\,.$$

□

Using the Iverson bracket notation we get:

$$\mathbb{E}\left[\,[\![\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k]\!]\,\right] = \Pr[\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k]\,.$$

Thus, based on Lemma 5.1 we can construct an unbiased estimator:

$$\mathbb{E}\left[\frac{1}{m}\sum_{k=1}^{m}[\![\mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k]\!]\right] = \frac{|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w}{|\Omega|_w}\,.$$

Then, we can define an unbiased estimator of $|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w$ as the product of two unbiased estimators for

$$\frac{|\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w}{|\Omega|_w} \quad \text{and} \quad |\Omega|_w\,.$$

Namely, analogously to formula (23) for

$$m' = |\{k : \mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k\}|$$

we have

$$\mathbb{E}\left[\frac{m'}{m}\,\frac{m-1}{\sum_{k=1}^{m} m_k}\right] = |\mathbb{A}\mathbb{B}\overline{\mathbb{C}}|_w\,. \tag{28}$$

Note that the two estimators don't have to be independent and to formally justify equation (28) we need to show that for any $x \geq 0$

$$\Pr[m_k \leq x \mid \mathbf{A}_k = \mathbf{B}_k < \mathbf{C}_k] = \Pr[m_k \leq x]\,. \tag{29}$$

We skip the proof of equation (29) as the line of reasoning is identical to the proof of Lemma 4.2 in [18].

To derive an estimator for $|\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w$ we use the following lemma.

Lemma 5.2.

$$\Pr[\mathbf{A}_k < \min\{\mathbf{B}_k, \mathbf{C}_k\}] = \frac{|\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w}{|\Omega|_w}\,. \tag{30}$$

Proof. The proof is very similar to the proof of Lemma 5.1. We only need to observe that

$$\Pr[\mathbf{A}_k < \min\{\mathbf{B}_k, \mathbf{C}_k\}] = \Pr\left[\min\left\{\mathbb{S}_\mathbb{A}\overline{\mathbb{S}}_\mathbb{B}\overline{\mathbb{S}}_\mathbb{C}\right\} < \min\left\{\Omega_\mathbb{S} \setminus \mathbb{S}_\mathbb{A}\overline{\mathbb{S}}_\mathbb{B}\overline{\mathbb{S}}_\mathbb{C}\right\}\right]$$

and thus that

$$\Pr[\mathbf{A}_k < \min\{\mathbf{B}_k, \mathbf{C}_k\}] = \frac{|\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w}{|\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w + |\Omega \setminus \mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w}\,.$$

□

Based on Lemma 5.2 we define unbiased estimator for $|\mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}|_w$ by the formula (28), but in this case

$$m' = |\{k : \mathbf{A}_k < \min\{\mathbf{B}_k, \mathbf{C}_k\}\}|\,.$$

In the same way we derive estimator for any other intersection. The correspondence between sets intersections and the appropriate formula for parameter $m'$ is given in Figure 2.

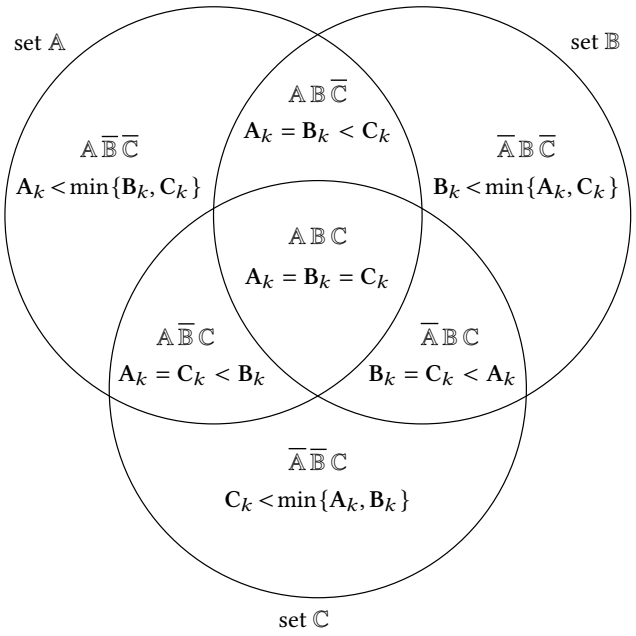



**Figure 2: Venn diagram representing correspondence between sets intersections and expressions required to compute the value of parameter $m'$ used in estimator (28) .**

## 5.3 General scheme

The scheme presented in the previous section can be generalized for any number of sets. For some $d \geq 2$ let us consider sets $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_d$ that constitute the universe $\Omega = \mathbb{A}_1 \cup \mathbb{A}_2 \cup \ldots \cup \mathbb{A}_d$ . Any sequence of set-theory operations on these sets could be transformed to a full disjunctive normal form – an alternative of disjoint intersections, each consisted of $d$ sets (or their complements). Let us consider one of such intersections. Since the intersection operation is commutative, without loss of generality we can assume that it has a form

$$\mathbb{A}_1 \ldots \mathbb{A}_r \overline{\mathbb{A}}_{r+1} \ldots \overline{\mathbb{A}}_d ,$$

where $r \geq 1$. Note that for $r = 0$ we get $\overline{\mathbb{A}}_1 \ldots \overline{\mathbb{A}}_d = \emptyset$ .

Let $\mathbf{A}_{i,k}$ be the $k$-th position in sketch $\mathbf{A}_i$ that represents set $\mathbb{A}_i$. Then we can prove the following lemma.

Lemma 5.3.

$$\Pr\left[\mathbf{A}_{1,k} = \ldots = \mathbf{A}_{r,k} < \min\{\mathbf{A}_{r+1,k}, \ldots, \mathbf{A}_{d,k}\}\right] = \frac{|\mathbb{A}_1 \ldots \mathbb{A}_r \overline{\mathbb{A}}_{r+1} \ldots \overline{\mathbb{A}}_d|_w}{|\Omega|_w} .$$

Proof. The proof is very similar to the proof of Lemma 5.1. We only need to observe that for set $\mathbb{Y} = \mathbb{S}_{\mathbb{A}_1} \ldots \mathbb{S}_{\mathbb{A}_r} \overline{\mathbb{S}}_{\mathbb{A}_{r+1}} \ldots \overline{\mathbb{S}}_{\mathbb{A}_d}$ we have

$$\Pr\left[\mathbf{A}_{1,k} = \ldots = \mathbf{A}_{r,k} < \min\{\mathbf{A}_{r+1,k}, \ldots, \mathbf{A}_{d,k}\}\right] = \Pr\left[\min\{\mathbb{Y}\} < \min\{\overline{\mathbb{Y}}\}\right] .$$

□

Therefore, analogously to formula (28), for

$$m' = |\{k : \mathbf{A}_{1,k} = \ldots = \mathbf{A}_{r,k} < \min\{\mathbf{A}_{r+1,k}, \ldots, \mathbf{A}_{d,k}\}\}|$$

and

$$m_k = \min\{\mathbf{A}_{1,k}, \ldots, \mathbf{A}_{d,k}\}$$

we get an unbiased estimator:

$$\mathbb{E}\left[\frac{m'}{m}\,\frac{m-1}{\sum_{k=1}^{m} m_k}\right] = |\mathbb{A}_1 \ldots \mathbb{A}_r \overline{\mathbb{A}}_{r+1} \ldots \overline{\mathbb{A}}_d|_w . \tag{31}$$

We can also easily justify that the variance of the above estimator can be expressed by the formula (21) with parameters

$$p = |\mathbb{A}_1 \ldots \mathbb{A}_r \overline{\mathbb{A}}_{r+1} \ldots \overline{\mathbb{A}}_d|_w \quad \text{and} \quad s = |\Omega|_w . \tag{32}$$

Namely, we can treat

$$\frac{m'}{m} \quad \text{and} \quad \frac{m-1}{\sum_{k=1}^{m} m_k}$$

as independent random variables (cf. Lemma 4.2 in [18]) and use the fact that if $X$ and $Y$ are independent random variables, then

$$\mathbb{V}\mathrm{ar}\,[XY] = \mathbb{E}[X]^2\mathbb{V}\mathrm{ar}\,[Y] + \mathbb{E}[Y]^2\mathbb{V}\mathrm{ar}\,[X] + \mathbb{V}\mathrm{ar}\,[X]\,\mathbb{V}\mathrm{ar}\,[Y] . \tag{33}$$

Expected value and variance of estimator $m'/m$ can be computed based on the fact that $m'$ is the sum of indicator random variables

$$m' = \sum_{k=1}^{m} [\![\mathbf{A}_{1,k} = \ldots = \mathbf{A}_{r,k} < \min\{\mathbf{A}_{r+1,k}, \ldots, \mathbf{A}_{d,k}\}]\!] .$$

From that we can infer that $m'$ must have the binomial distribution. Therefore, using Lemma 5.3 and notation given by equations (32) we get

$$\mathbb{E}\left[\frac{m'}{m}\right] = \frac{p}{s} \quad \text{and} \quad \mathbb{V}\mathrm{ar}\left[\frac{m'}{m}\right] = \frac{1}{m}\,\frac{p}{s}\left(1 - \frac{p}{s}\right) . \tag{34}$$

From equation (4) we know that $(m-1)/\sum_{k=1}^{m} m_k$ is an unbiased estimator. We also know that its variance is given by formula (5). Thus, by using the notation given by equations (32) we get that

$$\mathbb{E}\left[\frac{m-1}{\sum_{k=1}^{m} m_k}\right] = s \quad \text{and} \quad \mathbb{V}\mathrm{ar}\left[\frac{m-1}{\sum_{k=1}^{m} m_k}\right] = \frac{s^2}{m-2} . \tag{35}$$

Using formulas (33), (34) and (35) we obtain formula (21).

To sum up, a general scheme for estimating the weighted cardinality of any set $\mathbb{X}$ constructed with the use of set-theory operations on sets $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_d$ can be described as follows.

(1) Create sketches $\mathbf{A}_1, \ldots, \mathbf{A}_d$ representing sets $\mathbb{A}_1, \ldots, \mathbb{A}_d$.
(2) Express set $\mathbb{X}$ with the use of sets $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_d$ in a full disjunctive normal form to obtain an alternative of $b$ disjoint intersections for some $b \geq 1$.
(3) Assume $i$-th intersection has the form $\mathbb{A}_1 \ldots \mathbb{A}_r \overline{\mathbb{A}}_{r+1} \ldots \overline{\mathbb{A}}_d$. For $i \in \{1, 2, \ldots, b\}$ compute the value of
$$m'_i = \sum_{k=1}^{m} [\![\mathbf{A}_{1,k} = \ldots = \mathbf{A}_{r,k} < \min\{\mathbf{A}_{r+1,k}, \ldots, \mathbf{A}_{d,k}\}]\!] .$$
(4) Estimate $|\mathbb{X}|_w$ by summing up all the unbiased estimates for disjoint intersections based on formula (31)
$$\mathbb{E}\left[\frac{m'_1 + m'_2 + \ldots + m'_b}{m}\,\frac{m-1}{\sum_{k=1}^{m} m_k}\right] = |\mathbb{X}|_w . \tag{36}$$

(5) Since $m' = m'_1 + m'_2 + \ldots + m'_b$ is a random variable with the binomial distribution and

$$\mathbb{E}\left[\frac{m'}{m}\right] = \frac{|\mathbb{X}|_w}{|\Omega|_w}$$

by using formula (21) with $p = |\mathbb{X}|_w$ and $s = |\Omega|_w$ we get

$$\mathbb{V}\mathrm{ar}\left[\frac{m'}{m}\ \frac{m-1}{\sum_{k=1}^{m} m_k}\right] \approx \frac{|\mathbb{X}|_w\,|\Omega|_w}{m}\ . \tag{37}$$

(6) For $X$ being an estimator of $x$ and $\sigma[X] = \sqrt{\mathbb{V}\mathrm{ar}\,[X]}$ being its standard deviation, we define a relative standard error as $\mathbb{SE}\,[X] = \sigma[X/x]$ . Thus, by using formula (37) we get

$$\mathbb{SE}\left[\frac{m'}{m}\ \frac{m-1}{\sum_{k=1}^{m} m_k}\right] \approx \sqrt{\frac{|\Omega|_w}{m\ |\mathbb{X}|_w}}\ . \tag{38}$$

## 5.4 Conversion to FDNF

The issue that remains to be clarified is how to find a full disjunctive normal form (abbreviated as FDNF) in step (2) of the above scheme. An intuitive approach is to consider all $2^d$ possible intersections of length $d$ of sets $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_d$ and their complements and decide which are subsets of set $\mathbb{X}$. This approach is analogous to creating truth tables and converting them into FDNF boolean expressions. Obviously, it could be used only if the value of parameter $d$ is small.

The following syntactic approach (see [9]) is more convenient, but also carries the risk of the time complexity exponential in $d$.

(1) Use the fact that $\mathbb{A} \setminus \mathbb{B} = \mathbb{A} \cap \overline{\mathbb{B}}$ and de Morgan's laws to reduce an original expression to the form consisted only of sets $\mathbb{A}_i$ and $\overline{\mathbb{A}}_i$ combined by the unions and intersections (i.e. push negations down to the leaves of the syntax tree).
(2) Use the distributive law $\mathbb{A} \cap (\mathbb{B} \cup \mathbb{C}) = (\mathbb{A} \cap \mathbb{B}) \cup (\mathbb{A} \cap \mathbb{C})$ repeatedly, to obtain a union of intersections.
(3) The requirement that each set $\mathbb{A}_i$ occurs, either negated or not, only once in each intersection is achieved by dropping any intersections containing both $\mathbb{A}_i$ and $\overline{\mathbb{A}}_i$ , for any $i$. If neither $\mathbb{A}_i$ nor $\overline{\mathbb{A}}_i$ occurs in an intersection $\mathbb{I}$ , note that

$$\mathbb{I} = \mathbb{I} \cap (\mathbb{A}_i \cup \overline{\mathbb{A}}_i) = (\mathbb{I} \cap \mathbb{A}_i) \cup (\mathbb{I} \cap \overline{\mathbb{A}}_i)\ . \tag{39}$$

Let us remark that performing the step described by formula (39) might be unnecessary from the perspective of the estimator (36). Namely, to be able to simply add all estimates $m'_1 + m'_2 + \ldots m'_b$ we only require that all $b$ intersections are disjoint. For example, by joining two first intersections in the formula (26) we get

$$(\mathbb{A} \setminus \mathbb{C}) \ \cup\ \mathbb{A}\mathbb{B}\mathbb{C} \ =\ \mathbb{A}\mathbb{B} \ \cup\ \mathbb{A}\overline{\mathbb{B}}\,\overline{\mathbb{C}}\,. \tag{40}$$

Since $\mathbb{A}\,\mathbb{B}$ and $\mathbb{A}\,\overline{\mathbb{B}}\,\overline{\mathbb{C}}$ are disjoint, to use estimator (36) we can simply add

$$m'_1 = \sum_{k=1}^{m} [\![\mathbf{A}_k = \mathbf{B}_k]\!] \quad \text{and} \quad m'_2 = \sum_{k=1}^{m} [\![\mathbf{A}_k < \min\{\mathbf{B}_k, \mathbf{C}_k\}]\!]\ .$$

The form that is built as the union of disjoint intersections is called in the literature a disjoint disjunctive normal form (DDNF). A minimal DDNF formula can be obtained by Quine–McCluskey algorithm (see [21]). Let us remark that obtaining a minimal DDNF formula can be computationally costly (it's an NP-complete task). On the other hand, it may significantly reduce the computational cost of the considered estimation process.

# 6 EXPERIMENTS

In this section we present the experimental results. We implemented ExpSketch algorithm proposed in [18] (see: Algorithm 1) and newly proposed FastExpSketch algorithm (see: Algorithm 2) in the Wolfram Mathematica language. We make our implementation and experiments public (see: PVLDB Artifact Availability). They can be run in Wolfram Mathematica or in Wolfram Player, which is available for free. All results presented in this section were obtained on MacBook Air (1,6 GHz Intel Core i5, 16 GB 2133 MHz).

## 6.1 Verification of analytical results

In this section we verify analytical results presented in previous sections. Since the order of elements in a stream determines the number of comparisons we consider four representative scenarios:

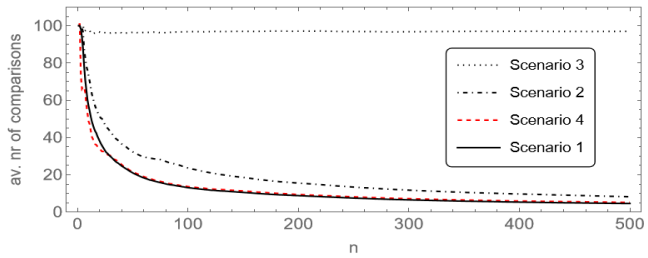
(1) $\lambda_1 = \lambda_2 = \ldots = \lambda_n$ ,
(2) $\lambda_i = i$ for $i \in \{1, 2, \ldots, n\}$ ,
(3) $\lambda_i = 2^i$ for $i \in \{1, 2, \ldots, n\}$ ,
(4) $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a random permutation of $\{1, 2, \ldots, n\}$ .

Scenarios (1) - (3) were already presented and analytically examined in Section 4.3. In Scenario (4) we consider a random arrangement. For the sake of figures readability, instead of the total number of comparisons we use average number of comparisons per element.

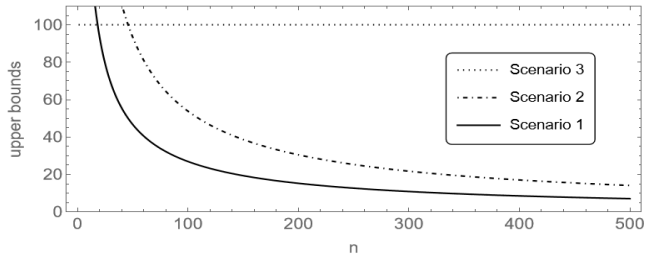
In Figure 3a we present the average number of comparisons per element for FastExpSketch in scenarios (1) – (4) when $m = 100$ and the number of stream elements $n$ is changing from 1 to 500. As expected by the analytical analysis, in scenario (3) we always have $m \approx 100$ comparisons per element and the average number of comparisons in scenario (2) is asymptoticly approximately two times larger than in scenario (1). In scenario (4) the values of weights are the same as in scenario (2) but they are arranged in random order. One can observe that for scenario (4) the number of comparisons is almost the same as in the most optimistic scenario (1). This confirms that scenarios in which the order of elements is ascending are pessimistic and in the average case the number of comparisons should be similar to this in scenario (1). Note that in ExpSketch we always have $m$ comparisons per each element.

In Figure 3b we present the upper bounds for the average number of comparisons per element for FastExpSketch in scenarios (1) - (3) when $m = 100$ and the number of stream elements $n$ is changing from 1 to 500. The upper bounds are derived from the analytical formulas obtained in Sections 4.3. When comparing the graphs to the corresponding graphs in Figure 3a we see that these upper bounds are quite strict.

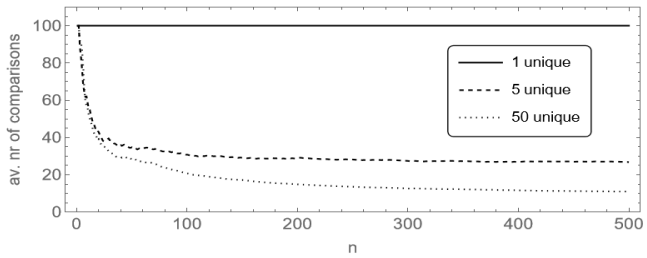
In case of FastExpSketch repetitions of elements also may have some impact on the computational complexity (see Section 4.4). Therefore, we analyze how the number of repeating elements in a stream affects the average number of comparisons per element. In Figure 3c we present the average number of comparisons per element for FastExpSketch when $m = 100$ and the number of stream elements $n$ changes from 1 to 500 but there are many copies of an element in the stream. We consider three cases. Namely, we take a set of $j$ elements $\Omega = \{(1, 1), (2, 2), \ldots, (j, j)\}$ for $j = 1, 5, 50$ and copy each element of this set 500, 100 and 10 times, respectively, to obtain 500 elements in total. To create a stream of $n$ elements, we take first $n$ values in a random permutation of these 500 elements.



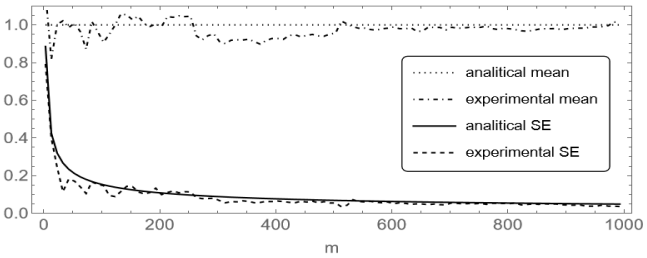

**(a) The average number of comparisons per element in different scenarios for $m = 100$ as $n$ changes from 1 to 500.**




**(b) Analytical upper bounds for the average number of comparisons per element for $m = 100$ as $n$ changes from 1 to 500.**




**(c) The average number of comparisons per element when there is 1, 5 and 50 unique elements in a stream and $m = 100$.**




**(d) Comparisons of analytical and experimental results on set given by formula (26) as $m$ changes from 1 to 1000 .**

**Figure 3: Experimental verification of analytical results for FastExpSketch algorithm.**

The experiment shows that if there is only one unique element in a stream, for each copy of this element we need $m = 100$ comparisons. However, as we mentioned in Section 4.4, when the number of unique elements grows, the average number of comparisons per element drops. For 50 unique elements the results are already similar to the results for scenario (4) in Figure 3a, in which we have a random permutation of the unique elements.

To experimentally verify the results of Section 5 we have defined the set of all possible elements as $\Omega = \{(1,1),(2,2),\ldots,(100,100)\}$ and constructed sets $\mathbb{A}, \mathbb{B}$ and $\mathbb{C}$ to be random subsets of $\Omega$ such that $|\mathbb{A}| = |\mathbb{B}| = |\mathbb{C}| = 50$ . Then, by using FastExpSketch for all three sets, we have constructed sketches **A**, **B** and **C**, respectively, for all values of parameter $m$ from 3 to 1000. Then we use these sketches and estimator defined by formula (36) to estimate the weighted cardinality of the set defined by formula (26). We repeat this experiment 10 times for each value of $m$ to approximate the mean and standard error of the estimator. In each experiment we use a different seed provided to the hash function. In Figure 3d we present the results of this experiment. By comparing the experimental and analytical results we can conclude that estimator (36) is actually unbiased and formula (38) for the standard error is correct.

## 6.2 Twitter dataset

In this and next sections we show that proposed sketches are suitable for online processing of massive data streams and enable time and memory efficient offline analysis. First, we analyze a stream of geotagged tweets from Twitter generated in Europe in April 6-8 2016 (see [20]). The entire stream contains 468 709 tweets and takes 1 GB of memory. In experiments we use user identifier (element id), user number of followers (element weight) and additionally user geolocalization, tweet language and time of tweet publication. We create 30 data sketches containing information about the distinct number of users and the average number of followers in the entire stream ($\Omega$), in different times of a day, countries and languages. We use sketches to run more precise queries, for example concerning users posting only in the afternoon, never posting in English and users from France or Spain posting only in English (see: Table 1).

In order to obtain the desired precision of estimates for queries on many sketches, we need to properly set the size $m$ of a sketch. For this purpose we use standard error formula (38) expressing how much in percent an estimate differs from the exact value on average. Assume that we aim at a standard error not greater than 10% and that for a set $\mathbb{X}$ being analyzed we have $|\mathbb{X}|_w / |\Omega|_w \approx 1/10$. Then, by formula (38) for $m = 10^3$ we get $\mathbb{SE} \approx 10\%$.

For $m = 10^3$ FastExpSketch needed **6 minutes** to generate all 30 sketches. ExpSketch for the same task needed **157 minutes**. All sketches take in total $30 \times m \times 32$ bits $= 0.12$ MB of memory. Note that all sketches are generated independently, so they can be generated in parallel 30 times faster. The code and results of presented experiments are available in Mathematica notebooks *Twitter_ExpSketch.nb* and *Twitter_FastExpSketch.nb*.

Based on sketches created by FastExpSketch for $m = 10^3$ we obtain the results in Table 1. For each set, we give the exact number of distinct users and the exact average number of followers per user. Below each exact number we give the standard error for an estimate based on sketches (expressed in percent). The average number of followers in a set is estimated as the estimate of the sum of followers for distinct users ($\lambda_i$ = nr of followers) divided by the estimate of the number of distinct users ($\lambda_i = 1$). Note that the standard error is at the level of 10% or lower in all cases except the last query, where the estimate for the average number of followers has the standard error equal to 28.3%. This is due to the fact that in this

**Table 1: Number of distinct users and average number of followers for different sets. For each exact number, we give the standard error of the estimate based on sketches ($m = 10^3$).**

| | Ω | Morning | Afternoon | Evening | Night |
|---|---|---|---|---|---|
| distinct users | 171 890 | 67 445 | 89 609 | 68 445 | 17 694 |
| $\mathbb{SE}$ (%) | 0.6 | 0.1 | 2.5 | 0.2 | 1.9 |
| av. followers | 1 535 | 1 591 | 1 613 | 1 466 | 1 578 |
| $\mathbb{SE}$ (%) | 3.3 | 0.9 | 7.2 | 2.1 | 4.9 |

| | UK | Spain | Germany | France | Poland |
|---|---|---|---|---|---|
| distinct users | 23 076 | 20 141 | 5 042 | 7 791 | 1 103 |
| $\mathbb{SE}$ (%) | 3.1 | 0.2 | 1.6 | 3.6 | 0.8 |
| av. followers | 2 340 | 1 514 | 1 938 | 2 735 | 1 111 |
| $\mathbb{SE}$ (%) | 0.3 | 2.0 | 3.6 | 10.1 | 4.8 |

| | English | Spanish | German | French | Polish |
|---|---|---|---|---|---|
| distinct users | 60 216 | 16 779 | 4 078 | 6 355 | 1 110 |
| $\mathbb{SE}$ (%) | 6.1 | 1.5 | 2.4 | 0.9 | 1.4 |
| av. followers | 1 869 | 1 445 | 1 426 | 2 672 | 821 |
| $\mathbb{SE}$ (%) | 7.9 | 1.3 | 1.8 | 2.2 | 0.3 |

| Afternoon \ (Morning ∪ Evening ∪ Night) | | | |
|---|---|---|---|
| distinct users | 53 768 | av. followers | 1 755 |
| $\mathbb{SE}$ (%) | 12.3 | $\mathbb{SE}$ (%) | 6.1 |

| (Spanish ∪ German ∪ French ∪ Polish) \ English | | | |
|---|---|---|---|
| distinct users | 24 144 | av. followers | 1 588 |
| $\mathbb{SE}$ (%) | 5.1 | $\mathbb{SE}$ (%) | 8.1 |

| (France ∪ Spain) ∩ (English \ (French ∪ Spanish)) | | | |
|---|---|---|---|
| distinct users | 1 965 | av. followers | 1380 |
| $\mathbb{SE}$ (%) | 9.4 | $\mathbb{SE}$ (%) | **28.3** |

case $|\mathbb{X}|_w / |\Omega|_w \approx 2/100$ and this ratio is too small for $m = 10^3$. To increase the granularity of the query we need to increase $m$. For example, if $|\mathbb{X}|_w / |\Omega|_w \approx 1/100$ and we aim at $\mathbb{SE} \approx 10\%$, we need to set $m = 10^4$ . In practice, we can use a rough initial estimate of $|\mathbb{X}|_w / |\Omega|_w$ to adjust $m$. The time of performing queries and operations on sketches is negligible (given in milliseconds).

## 6.3 Twitter API

To test FastExpSketch in the streaming scenario we created a notebook *Twitter_API.nb* that allows to connect to Twitter API and build 30 sketches analogous to these in Section 6.2. Due to Twitter's policy, we were able to fetch only 6 million tweets in $6 \cdot 10^4$ batches, each batch containing approximately 100 tweets. Most of the time was spent waiting for the API to respond (also Twitter's policy).

In Table 2 we analyze the time of experiment broken down to different activities for $m = 10^3$. Waiting for the API and fetching data take almost 97% of the time. Updating a sketch takes 1.03 hour, which does not exceed 3% of the time. On average, updating a sketch by a single batch took $0.06[s]$. The relatively large standard deviation from the mean ($0.05[s]$) results from the fact that initial batches consume much more time than later ones (cf. Section 6.1).

Estimates based on created sketches are shown in *Twitter_API.nb*. Their accuracy is similar to accuracy of estimates from Section 6.2. Due to Twitter's policy, we cannot share the obtained dataset, however, we share a file with *ids* of collected tweets and the code that allows to fetch them (so called *hydrating tweets*).

**Table 2: Building 30 sketches for 6 million tweets ($m = 10^3$). For each activity we show total time in hours and percentage, mean time and standard deviation for a single batch [sec].**

| | Total [h] | Total [%] | Mean [s] | SD [s] |
|---|---|---|---|---|
| Waiting for API | 22.22 | 58.55 | 1.19 | 10.8 |
| Fetching data | 13.23 | 38.32 | 0.78 | 0.13 |
| Processing data | 0.04 | 0.13 | 0.002 | 0.0005 |
| Updating sketch | 1.03 | 2.98 | 0.06 | 0.05 |

## 6.4 Stream simulator

Since the Twitter API did not allow to fully show the efficiency of FastExpSketch (too little data coming in too slowly), in notebook *simulator.nb* we share the code for modeling a data stream. It allows to specify elements in a stream, the batch size and the time of fetching a batch. The latter can be defined by a probability distribution.

In the notebook we run three trials for $n = 10^9$ elements and the batch size equal to 1000. Since the fetching time can be modeled arbitrarily, in Table 3 we only report times for the sketch updates.

Let us compare the first two rows of Table 3. We can say that the total time cost of increasing $m$ from $10^3$ to $10^4$ is not very significant. By comparing the mean and maximal time for a single batch one can deduce that the substantial difference is only for initial batches.

By comparing the second and third row of Table 3 one can also say that in a pessimistic scenario, in which weights of elements gradually grow, there is no significant increase of the total time cost. As previously, the substantial difference is only for initial batches.

**Table 3: Sketching $10^9$ elements $(i, \lambda_i)$ with FastExpSketch. For given $m$ and $\lambda_i$ we show total time of sketch updates in hours and mean and maximal time for a single batch [sec].**

| $m$ | $\lambda_i$ | Total [h] | Mean [s] | Max [s] |
|---|---|---|---|---|
| $10^3$ | rand(0,1) | 4.33 | 0.016 | 1.127 |
| $10^4$ | rand(0,1) | 5.45 | 0.020 | 30.112 |
| $10^4$ | $i$ | 5.79 | 0.021 | 61.609 |

# 7 CONCLUSIONS

Our three main contributions form a complete solution.

**Contribution 1:** Proof that estimator $\overline{\Lambda} = \frac{m-1}{G_m}$ defined in [18] is a maximum likelihood estimator and thus is asymptotically most efficient and normally distributed when $m$, the number of experiments, increases to infinity. Showing that to estimate the weighted cardinality of any set constructed as a sequence of set-theory operations with the desired level of precision, one need to choose the appropriate value of parameter $m$ according to formula (38).

**Contribution 2:** Much faster algorithm for updating a sketch. The idea is to reduce the number of hashes that are computed for each element from $m$ to a constant by considering them in increasing order and having an early stopping criterion.

**Contribution 3:** Generalization of the solution of [18] allowing simple set operations to a solution allowing arbitrary set operations. The idea is to transform a set-theory expression to a full disjunctive normal form consisting of disjoint set intersections, estimate every intersection separately, and then sum up all these estimates. We believe that is the most important contribution of this paper.

# REFERENCES


[1] Milton Abramowitz and Irene A. Stegun. 1964. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, New York.

[2] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable Summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Scottsdale, Arizona, USA) *(PODS '12)*. Association for Computing Machinery, New York, NY, USA, 23–34. https://doi.org/10.1145/2213556.2213562

[3] Carlos Baquero, Paulo Sérgio Almeida, and Raquel Menezes. 2009. Fast Estimation of Aggregates in Unstructured Networks. In *Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems*. IEEE Computer Society, 88–93.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting Distinct Elements in a Data Stream. In *RANDOM*. 1–10.

[5] Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis. 2009. Distinct-Value Synopses for Multiset Operations. *Commun. ACM* 52, 10 (Oct. 2009), 87–95. https://doi.org/10.1145/1562764.1562787

[6] Philippe Chassaing and Lucas Gerin. 2006. Efficient estimation of the cardinality of large data sets. In *4th Colloquium on Mathematics and Computer Science*. DMTCS Proceedings, 419–422.

[7] Reuven Cohen, Liran Katzir, and Aviv Yehezkel. 2015. A Unified Scheme for Generalizing Cardinality Estimators to Sum Aggregation. *Inf. Process. Lett.* 115, 2 (Feb. 2015), 336–342. https://doi.org/10.1016/j.ipl.2014.10.009

[8] Anirban Dasgupta, Kevin J. Lang, Lee Rhodes, and Justin Thaler. 2016. A Framework for Estimating Stream Expression Cardinalities. In *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016 (LIPIcs)*, Wim Martens and Thomas Zeume (Eds.), Vol. 48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:17. https://doi.org/10.4230/LIPIcs.ICDT.2016.6

[9] B.A. Davey and H.A. Priestley. 2002. *Introduction to Lattices and Order*. Cambridge University Press.

[10] Luc Devroye. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, NY, USA.

[11] Nick Duffield, Carsten Lund, and Mikkel Thorup. 2007. Priority Sampling for Estimation of Arbitrary Subset Sums. *J. ACM* 54, 6 (Dec. 2007), 32–es.

[12] M. Durand and P. Flajolet. 2003. Loglog Counting of Large Cardinalities. In *Annual European Symposium on Algorithms (ESA03) (Lecture Notes in Computer Science)*, G. Di Battista and U. Zwick (Eds.), Vol. 2832. Springer Berlin Heidelberg, 605–617.

[13] Otmar Ertl. 2017. New Cardinality Estimation Methods for HyperLogLog Sketches. *CoRR* (2017). arXiv:1706.07290 http://arxiv.org/abs/1706.07290

[14] Otmar Ertl. 2021. SetSketch: Filling the Gap between MinHash and HyperLogLog. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2244–2257. https://doi.org/10.14778/3476249.3476276

[15] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. 2007. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the Conference on Analysis of Algorithms (AofA'07)*. 127–146.

[16] Stefan Heule, Marc Nunkesser, and Alex Hall. 2013. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. In *Proceedings of the EDBT 2013 Conference*. Association for Computing Machinery, Genoa, Italy, 683–692.

[17] Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (third ed.). Addison-Wesley, Boston.

[18] Jakub Lemiesz. 2021. On the algebra of data sketches. *Proc. VLDB Endow.* 14, 9 (2021), 1655–1667. https://doi.org/10.14778/3461535.3461553

[19] Damon Mosk-Aoyama and Devavrat Shah. 2006. Computing separable functions via gossip. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing (PODC '06)*. Association for Computing Machinery, 113–122.

[20] Wolfram Research. 2016. Geotagged Public Tweets (Europe, April 6-8 2016). https://doi.org/10.24097/wolfram.73020.data

[21] W.G. Schneeweiss. 2012. *Boolean Functions: With Engineering Applications and Computer Programs*. Springer Berlin Heidelberg.

[22] Daniel Ting. 2016. Towards Optimal Cardinality Estimation of Unions and Intersections with Sketches. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 1195–1204. https://doi.org/10.1145/2939672.2939772