A Minimal Variance Estimator for the Cardinality of Big Data Set Intersection

Reuven Cohen
Department of Computer Science
Technion
Haifa 32000, Israel
rcohen@cs.technion.ac.il

Liran Katzir
Department of Computer Science
Technion
Haifa 32000, Israel

Aviv Yehezkel
Department of Computer Science
Technion
Haifa 32000, Israel
avivy@cs.technion.ac.il

ABSTRACT

In recent years there has been a growing interest in developing "streaming algorithms" for efficient processing and querying of continuous data streams. These algorithms seek to provide accurate results while minimizing the required storage and the processing time, at the price of a small inaccuracy in their output. A fundamental query of interest is the intersection size of two big data streams. This problem arises in many different application areas, such as network monitoring, database systems, data integration and information retrieval. In this paper we develop a new algorithm for this problem, based on the Maximum Likelihood (ML) method. We show that this algorithm outperforms all known schemes in terms of the estimation's quality (lower variance) and that it asymptotically achieves the optimal variance.

CCS CONCEPTS

• **Information systems →Data stream mining;**

KEYWORDS

Data mining; Streaming algorithms; Cardinality estimation; Set intersection

ACM Reference format:
Reuven Cohen, Liran Katzir, and Aviv Yehezkel. 2017. A Minimal Variance Estimator for the Cardinality of Big Data Set Intersection. In *Proceedings of KDD'17, August 13–17, 2017, Halifax, NS, Canada., ,* 9 pages.
DOI: http://dx.doi.org/10.1145/3097983.3097999

1 INTRODUCTION

Classical processing algorithms for database management systems usually require several passes over (static) data sets in order to produce an accurate answer to a user query. However, for a wide range of application domains, the data set is very large and is updated on a continuous basis, making this approach impractical. For this reason, there is a growing interest in developing "streaming algorithms" for efficient processing and querying of continuous data streams in data stream management systems (DSMSs). These algorithms seek to provide accurate results while minimizing both the required storage and the processing time per stream element, at the price of a small inaccuracy in their output [2, 7, 11, 20]. Streaming algorithms for DSMSs typically summarize the data stream using a small sketch, and use probabilistic techniques in order to provide approximate answers to user queries. Such big data streams appear in a wide variety of computer science applications. They are common, for example, in computer networks, where detailed usage statistics (such as the source IP addresses of packets) from different parts of the network need to be continuously collected and analyzed for various security and management tasks.

A fundamental query of interest is the intersection size of two big data streams. Consider two streams of elements, $\overline{A}$ and $\overline{B}$, taken from two sets A and B respectively. Suppose that each element may appear more than once in each stream. Let $n = \left|\overline{A} \cap \overline{B}\right|$. For example, for $\overline{A} = a, b, c, d, a, b$ and $\overline{B} = a, a, c, c$, we get that $n = 2$ is the cardinality of $A \cap B$. Finding n is a problem that arises in many different application areas such as network monitoring, database systems, data integration and information retrieval [4, 5].

One can find the exact value of n by computing the intersection set C in the following way. For every element $b_i \in B$, compare b_i to every $a_j \in A$ and $c_k \in C$. If $b_i \notin C$ and $b_i \in A$, add b_i to C. After all the elements are treated, return the number of elements in C. This naive approach does not scale if storage is limited or if the sets are very large. In these cases, the following estimation problem should be solved. Given two streams of elements (with repetitions) $\overline{A} = a_1, a_2, \ldots, a_p$, and $\overline{B} = b_1, b_2, \ldots, b_q$, such that A and B are the respective sets of the two streams, and $n = |A \cap B|$, find an estimate $\widehat{n}$ of n using only m storage units, where $m \ll n$.

The cardinality $|A \cap B|$ can be estimated in a straightforward manner using the inclusion-exclusion principle: $\widehat{|A \cap B|} = \widehat{|A|} + \widehat{|B|} - \widehat{|A \cup B|}$, taking advantage of the fact that estimating $|A|$, $|B|$ and $|A \cup B|$ with good accuracy (small variance) is easy. However, we will later show that this scheme produces inaccurate results. In [2], it is proposed to estimate the Jaccard similarity, defined as $\rho(A, B) = \frac{|A \cap B|}{|A \cup B|}$. The idea is to estimate $|A \cup B|$, and then to extract $|A \cap B|$.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

DOI: http://dx.doi.org/10.1145/3097983.3097999

A third scheme, proposed in [11], suggests that $\widehat{|A \cap B|} = \frac{\widehat{\rho(A,B)}}{\rho(A,B)+1}(\widehat{|A|} + \widehat{|B|})$. With respect to the above three estimation schemes, the main contributions of this paper are as follows:

(1) **We present and analyze a new unbiased estimator, based on the Maximum Likelihood (ML) method, which outperforms the above three schemes by having a lower variance.**
(2) **The new estimator has the optimal (minimum) variance of any unbiased set intersection estimator. We present an analysis of this variance.**

There are several previous works on set intersection estimators (e.g., [2, 11, 24]), but none is based on the Maximum Likelihood (ML) estimator, which asymptotically achieves the optimal (lower bound) variance. The most relevant attempt is a very recent work [24], which describes a "pseudo-likelihood based method". This method is claimed by the authors to be asymptotically equivalent to a maximum likelihood estimator. In this paper, we derive, for the first time, the complete maximum likelihood estimator and provide a full proof.

The rest of the paper is organized as follows. Section 2 provides the motivation for this work and discusses a wide variety of applications for which accurate estimation of set intersection cardinality is required. Section 3 discusses previous work and presents the three previously known schemes. Section 4 presents our new Maximum Likelihood (ML) estimator. It also shows that the new scheme achieves optimal variance and that it outperforms the three known schemes. Section 5 presents simulation results confirming that the new ML estimator outperforms the three known schemes. Section 6 presents analysis results of the three previous schemes. Finally, Section 7 concludes the paper.

2 MOTIVATION AND APPLICATIONS

Many real world problems from different disciplines require estimation of the cardinality of set intersection. In this section we describe some of them.

Relational database management systems have achieved great success in the past forty years. One of the main reasons is that each system uses declarative query languages, such as SQL, which allow to describe a query without specifying how it should be executed. This abstraction has greatly benefited users and developers, who no longer need to understand how the data is actually stored and manipulated by these languages. At the same time, declarative query languages allow great flexibility in choosing different plans for a given query. A plan is a tree-like structure that encodes query sub-parts. Each node indicates some operator, such as scan, sort, or join. In order to produce efficient, low-cost, execution plans, every database system contains a query optimizer that can estimate the cost of each plan.

Query optimizers usually define the cost of a query according to its predicted CPU and I/O overhead. To estimate these quantities, an optimizer needs to estimate the input/output cardinalities of each operator in the plan. For example, in join and conjunctive queries, the order in which intersections are processed by the database system significantly affects the performance, and the optimizer needs to know the number of distinct tuples in each intersection in order to choose the best strategy. Thus, the query optimizer needs to accurately estimate set intersection cardinalities. In addition, by estimating the set intersection cardinality between all involved tables, the optimizer can estimate the size of the join/conjunction outcome.

Another discipline where estimating the set intersection cardinality is important is network management and security [14, 15, 20]. For example, web administrators would like to measure the increase in popularity of their sites, e.g., to estimate the number of users who accessed the web server during the last 24 hours, but not during the previous week. This can be translated into the problem of estimating the cardinality of $|V_t \setminus V_w| = |V_t| - |V_t \cap V_w|$, where V_t is the stream of users (e.g., their source IP addresses) who visited the site today, and V_w is the stream of users who visited the site during the previous week.

Estimating the set intersection cardinality also allows real-time detection of network and security anomalies. As an example, consider a network where packets are received from a router. Let R be the set of IP flows that enter the network through the router. We hereby describe two important monitoring applications:

(a) Suppose that all incoming flows must pass through a firewall. To verify this, one needs to verify that the set $R \cap F$ is equal to F, where F is the set of flows that traverse the firewall. This can be translated into the problem of estimating $|R \cap F|$ and verifying that it is "close enough" to $|F|$.
(b) Internet policy enforcement: The Internet is composed of many Autonomous Systems (AS). Two neighboring ASs can have a peering or a customer-provider relationship. The relationship determines which packets can be forwarded from one to the other. For example, an AS is not allowed to forward a packet to a peer AS if the ultimate destination of the packet is a client of the first AS. By estimating set intersection cardinality between the stream of packets received from each neighboring AS, an AS can discover routing violations and anomalies.

Our last application example is finding textually similar documents in a large corpus such as the web or a collection of news articles. In this case, $a_i \in A$ and $b_j \in B$ could be sets of sub-strings found within two text documents. Then, $|A \cap B|$ represents the number of sub-strings shared by both documents, and $\frac{|A \cap B|}{|A \cup B|}$ can be used as a "similarity index". Among many applications, this allows plagiarism detection and pruning of near-duplicate search results in search engines.

3 RELATED WORK AND PREVIOUS SCHEMES

The database research community has extensively explored the problem of data cleaning: detecting and removing errors and inconsistencies from data to improve the quality of databases [22]. Identifying which fields share similar values, identifying join paths, estimating join directions and sizes, and detecting inclusion dependencies are well-studied aspects of this problem [1, 8, 11, 17, 21]. For example, in [11] the authors present several methods for finding related database fields. Their main idea is to hash the values of each field and keep a small sketch that contains the minimal hash values for each. Then, the Jaccard similarity is used to measure similarities between fields. In [1], the authors study the related problem of detecting inclusion dependencies, i.e., pairs of fields A and B such that $A \subseteq B$, and they present an efficient way to test all field pairs in parallel.

3.1 The Cardinality Estimation Problem

Algorithms for estimating the cardinality of set intersection use estimations of $|A|$, $|B|$, and $|A \cup B|$. These estimations can be found using well-known algorithms for the following cardinality estimation problem:

Instance: A stream of elements $x_1, x_2, \ldots, x_s$ with repetitions. Let c be the number of different elements, namely $c = |\{x_1, x_2, \ldots, x_s\}|$.

Objective: Find an estimate $\widehat{c}$ of c using only m storage units, where $m \ll c$.

This problem has received a great deal of attention in the past decade thanks to the growing number of important real-time "big data" applications, such as estimating the propagation rate of viruses, detecting DDoS attacks [14, 15], and measuring general properties of network traffic [20].

Many works address the cardinality estimation problem [7, 9, 12, 16, 19, 20] and propose statistical algorithms for solving it. These algorithms are usually limited to performing only one pass on the received packets and using a fixed small amount of memory. A common approach is to hash every element into a low-dimensional data sketch, which can be viewed as a uniformly distributed random variable. Then, one of the following schemes is often used to estimate the number of distinct elements in the set:

(1) Order-statistics based estimators: In this family of schemes, the identities of the smallest (or largest) k elements are remembered for the considered set. These values are then used for estimating the total number of distinct elements [2, 9, 16, 19]. The family of estimators with $k = 1$ (where the minimal/maximal identity is remembered) is also known as min/max sketches.

(2) Bit-pattern based estimators: In this family of schemes, the highest position of the leftmost 1-bit in the binary representation of the identity of each element is remembered and then used for the estimation [7, 12].

If only one hash function is used, the schemes estimate the value of n with an infinite variance. To bound the variance, both schemes repeat the above procedures for m different hash functions and use their combined statistics for the estimation[1].

A comprehensive overview of different cardinality estimation techniques is given in [7, 20]. State-of-the art cardinality estimators have a standard error of about $1/\sqrt{m}$, where m is the number of storage units [6]. The best known cardinality estimator is the HyperLogLog algorithm [12], which belongs to the family of min/max sketches and has a standard error of $1.04/\sqrt{m}$, [12]. For instance, this algorithm estimates the cardinality of a set with 10^9 elements with a standard error of 2% using $m = 2,048$ storage units.

For the rest of the paper we consider the HyperLogLog algorithm [12] for solving the cardinality estimation problem. The estimator uses $O(1)$ operations per element, and is very efficient because it maintains the m sketches using only two hash functions, h_1 and h_2. The first function is used for hashing each element into a uniformly distributed variable (the element's sketch), and the second for splitting each element into one of m buckets. These m buckets replace the need for m different hash functions. In addition, in order to increase computation speed, the estimator does not keep the value of the maximal sketch for each bucket, but rather only the highest position of the leftmost 1-bit of each sketch i.e., the largest $i \geq 0$ such that the $i-1$ leftmost bits are all 0.

3.2 Previous Schemes

Cardinality estimation algorithms can be used for estimating the cardinality of set intersection. As mentioned in Section 1, a straightforward technique is to estimate the intersection using the following inclusion-exclusion principle:

$$\widehat{|A \cap B|} = \widehat{|A|} + \widehat{|B|} - \widehat{|A \cup B|}.$$

This method will be referred to as Scheme-1. Other set intersection estimators first estimate the Jaccard similarity $\rho(A,B) = \frac{|A \cap B|}{|A \cup B|}$. The value of ρ ranges between 0, when the two sets are completely different, and 1, when the sets are identical.

An efficient and accurate estimate of ρ can be computed as follows [3]. Each item in A and B is hashed into $(0,1)$, and the maximal value of each set is considered as a sketch that represents the whole set. To improve accuracy, m hash functions are used, and the sketch of each set is a vector of m maximal values. Given a set $A = \{a_1, a_2, \ldots, a_p\}$ and m different hash functions $h_1, h_2, \ldots, h_m$, the maximal hash value for the jth hash function can be formally expressed as:

$$x_A^j = \max_{i=1}^{p} \{h_j(a_i)\} \quad , \quad 1 \leq j \leq m,$$

and the sketch of A is:

$$X_{(A)} = \left\{x_A^1, x_A^2, \ldots, x_A^m\right\}.$$

[1] Stochastic averaging can be used to reduce the number of hash functions from m to only two [13].

$X_{(B)}$ is computed in the same way. Then, the two sketches are used to estimate the Jaccard similarity of A and B:

$$\widehat{\rho(A,B)} = \frac{\sum_{j=1}^{m} I_{x_A^j == x_B^j}}{m}, \quad (1)$$

where the indicator function $I_{x_A^j == x_B^j}$ is 1 if $x_A^j = x_B^j$, and 0 otherwise. To shorten our notation, for the rest of the paper we use ρ to indicate $\rho(A,B)$.

After estimating the Jaccard similarity, previous works use some algebraic manipulation on it in order to estimate the set intersection cardinality [2, 11]. Specifically, the scheme proposed in [2] estimates both the Jaccard similarity and $|A \cup B|$, and then uses

$$\widehat{|A \cap B|} = \widehat{\rho(A,B)} \cdot \widehat{|A \cup B|}.$$

This scheme will be referred to as Scheme-2. The third scheme discussed in this paper, referred to as Scheme-3, is presented in [11]. It estimates the Jaccard similarity, $|A|$, and $|B|$, and then uses

$$\widehat{|A \cap B|} = \frac{\widehat{\rho(A,B)}}{\rho(A,B)+1}(\widehat{|A|} + \widehat{|B|}).$$

The above equation is obtained by substituting the Jaccard similarity definition into $\frac{\rho(A,B)}{\rho(A,B)+1}$, which yields that

$$\frac{\rho(A,B)}{\rho(A,B)+1} = \frac{|A \cap B|}{|A \cap B| + |A \cup B|} = \frac{|A \cap B|}{|A| + |B|}.$$

In [1, 18], the set intersection estimation problem is solved using smaller sample sets. While these techniques are simple and unbiased, they are inaccurate for small sets. In addition, they are sensitive to the arrival order of the data, and to the repetition pattern.

Recent work [24] provides a "pseudo-likelihood based method" for estimating the set intersection cardinality. The method is claimed by the authors to be asymptotically equivalent to a maximum likelihood estimator, but there is no proof for this claim. In this paper, we derive the complete maximum likelihood estimator and provide a full proof.

4 A NEW MAXIMUM LIKELIHOOD SCHEME WITH OPTIMAL VARIANCE

In this section we present a new unbiased estimator for the set intersection estimation problem. Because this estimator is based on the Maximum Likelihood (ML) method, it achieves optimal variance and outperforms the three known schemes.

Maximum-Likelihood estimation (ML) is a method for estimating the parameters of a statistical model. For example, suppose we are interested in the height distribution of a given population, but are unable to measure the height of every single person. Assuming that the heights are Gaussian distributed with some unknown mean and variance, the mean and variance can be estimated using ML and only a small sample of the overall population. In general, for a given set of data samples and an underlying statistical model, ML finds the values of the model parameters that maximize the likelihood function, namely, the "agreement" of the selected model with the given sample.

In the new scheme, we first find the (probability density) likelihood function of the set intersection estimation problem, $L(x_A = s, x_B = t; \theta)$; namely, given $\theta = (a, b, n)$ as the problem parameters[2], this is the probability density of s to be the maximal hash value for A and t to be the maximal hash value for B. Then, we look for values of the problem parameters that maximize the likelihood function.

Table 1 shows some of the notations we use for the rest of the paper.

notation	value
n	$\lvert A \cap B \rvert$
u	$\lvert A \cup B \rvert$
a	$\lvert A \rvert$
b	$\lvert B \rvert$
α	$\lvert A \setminus B \rvert$
β	$\lvert B \setminus A \rvert$

Table 1: Notations

4.1 The Likelihood Function of the Set Intersection Estimation Problem

We first find the likelihood function for one hash function h_k, namely, $L(x_A = s, x_B = t; \theta)_k$, and then generalize it for all hash functions. Recall that for the kth hash function, $x_A^k = \max_{i=1}^{a} \{h_k(a_i)\}$ and $x_B^k = \max_{j=1}^{b} \{h_k(b_j)\}$. To simplify the notation, we shall omit the superscript k for x_A^k and x_B^k, and use x_A and x_B respectively.

We use $\mathrm{PDF}_U(w)$ to denote the probability density function (PDF) of a uniformly distributed random variable $\mathrm{U}(0,1)$ at w. Denote the elements in $A \cap B$ as $\{z_1, z_2, \ldots, z_n\}$. Thus, the elements in A and the elements in B can be written as $\{x_1, x_2, \ldots, x_\alpha, z_1, z_2, \ldots, z_n\}$ and $\{y_1, y_2, \ldots, y_\beta, z_1, z_2, \ldots, z_n\}$ respectively (see Table 1).

We now divide the likelihood function according to the three possible relations between x_A and x_B: $x_A = x_B$, $x_A > x_B$ and $x_A < x_B$.

Case 1: $x_A = x_B$

When $x_A = x_B = s$ holds, the element with the maximal hash value must belong to $A \cap B$. The likelihood function of θ given this outcome is

$$L(x_A = x_B = s; \theta) = \sum_{i=1}^{n} \mathrm{PDF}_U(s) \cdot \Pr\left(x_{(A \cup B) \setminus \{z_i\}} < s\right)$$
$$= \sum_{i=1}^{n} s^{u-1} = n \cdot s^{u-1}. \quad (2)$$

[2]The set intersection estimation problem has six identifying parameters $(n, u, a, b, \alpha, \beta)$, only three of which are needed to derive the others.

This equality holds because there are n possible elements in $A \cap B$ whose hash value can be the maximum in $A \cup B$, and because $\text{PDF}_U(s) = 1$.

Case 2: $x_A < x_B$

In order to have $x_A < x_B$, where $x_A = s$ and $x_B = t$, the maximal hash value in B must also be in $B \setminus A$, and its value must be t. The likelihood function of θ in this case is

$$L(x_{B\setminus A} = t; \theta) = \sum_{j=1}^{\beta} \text{PDF}_U(t) \cdot \Pr\left(x_{(B\setminus A)\setminus\{y_j\}} < t\right) = \beta \cdot t^{\beta-1}.$$

In addition, the maximal hash value in A must be s. The probability density for this is

$$L(x_A = s; \theta) = \sum_{e\in A} \text{PDF}_U(s) \cdot \Pr\left(x_{A\setminus\{e\}} < s\right) = as^{a-1}.$$

Thus,

$$L(x_A < x_B, x_A = s, x_B = t; \theta) = as^{a-1} \cdot \beta t^{\beta-1}. \quad (3)$$

Case 3: $x_A > x_B$

This case is symmetrical to the previous case. Thus, the likelihood function of θ in this case is

$$L(x_A > x_B, x_A = s, x_B = t; \theta) = \alpha s^{\alpha-1} bt^{b-1}.$$

Thus, the likelihood function for set intersection is

$$L(x_A = s, x_B = t; \theta)_k = \begin{cases} ns^{u-1} & x_A = x_B \\ \beta t^{\beta-1} as^{a-1} & x_A < x_B \\ \alpha s^{\alpha-1} bt^{b-1} & x_A > x_B \end{cases}.$$

We now use the following indicator variables:

(1) $I_1 = 1$ if $x_A = x_B$, and $I_1 = 0$ otherwise,
(2) $I_2 = 1$ if $x_A < x_B$, and $I_2 = 0$ otherwise,
(3) $I_3 = 1$ if $x_A > x_B$, and $I_3 = 0$ otherwise,

to obtain that

$$L(x_A = s, x_B = t; \theta)_k = (ns^{u-1})^{I_1} \cdot (\beta t^{\beta-1} as^{a-1})^{I_2} \cdot (\alpha s^{\alpha-1} bt^{b-1})^{I_3}. \quad (4)$$

Eq. (4) states the likelihood function for one hash function. To generalize this equation to all m hash functions, denote $S = (s_1, s_2, \ldots, s_m)$ and $T = (t_1, t_2, \ldots, t_m)$ as the m-dimensional vectors of s and t for each hash function.

Corollary 4.1.
The likelihood function for the set intersection estimation problem, for all m hash functions, satisfies:

$$L(x_A = S, x_B = T; \theta) = \prod_{k=1}^{m} \Big((n(s_k)^{u-1})^{I_{1,k}} \cdot (\beta(t_k)^{\beta-1} a(s_k)^{a-1})^{I_{2,k}} \cdot (\alpha(s_k)^{\alpha-1} b(t_k)^{b-1})^{I_{3,k}} \Big),$$

where $I_{1,k}$ is the value of I_1 for hash function k, and the same holds for $I_{2,k}$ and $I_{3,k}$. ■

It is usually easier to deal with the log of a likelihood function than with the likelihood function itself. Because the logarithm is a monotonically increasing function, its maximum value is obtained at the same point as the maximum of the function itself. In our case,

$$\begin{aligned} \log L(x_A = S, x_B = T; \theta) &= \log \prod_{k=1}^{m} L(x_A^k = s_k, x_B^k = t_k; \theta)_k \\ &= \sum_{k=1}^{m} \log L(x_A^k = s_k, x_B^k = t_k; \theta)_k \\ &= \sum_{k=1}^{m} I_{1,k} \cdot \log\left(n \cdot (s_k)^{u-1}\right) \\ &+ \sum_{k=1}^{m} I_{2,k} \cdot \log\left(\beta \cdot (t_k)^{\beta-1} \cdot a \cdot (s_k)^{a-1}\right) \\ &+ \sum_{k=1}^{m} I_{3,k} \cdot \log\left(\alpha \cdot (s_k)^{\alpha-1} \cdot b \cdot (t_k)^{b-1}\right). \end{aligned} \quad (5)$$

4.2 The New Scheme

We use Corollary 4.1 in order to find $\theta = (a, b, n)$ that maximizes Eq. (5). Note that there must be a single global optimum to Eq. (5) because this function is concave (sum of linear and logarithmic functions). Let $g(a, b, n)$ and $\mathbb{H}(a, b, n)$ be the gradient and the Hessian matrix of the log-likelihood function. Namely, g is the vector whose components are the partial derivatives of the log-likelihood function for the problem parameters $\theta = (a, b, n)$:

$$g(a, b, n) = \left(\frac{\partial \log L}{\partial a}, \frac{\partial \log L}{\partial b}, \frac{\partial \log L}{\partial n}\right), \quad (6)$$

and $\mathbb{H}$ is the matrix of the second-order partial derivatives of the log-likelihood function

$$\mathbb{H}_{i,j} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log L \quad , 1 \le i, j \le 3\,, \quad (7)$$

where $\theta_1 = a$, $\theta_2 = b$ and $\theta_3 = n$.

The new scheme finds the maximum of the log-likelihood function, i.e., the root of its gradient $g(a, b, n)$. Practically, this is done using iterations of the Newton-Raphson method on the gradient and Hessian matrix g and H. Starting from an initial estimation $\widehat{\theta_0} = (\widehat{a_0}, \widehat{b_0}, \widehat{n_0})$, the Newton-Raphson method implies that a better estimation is $\widehat{\theta_1} = \widehat{\theta_0} - H^{-1}(\widehat{\theta_0}) \cdot g(\widehat{\theta_0})$. The process is repeated, namely,

$$\widehat{\theta_{l+1}} = \widehat{\theta_l} - H^{-1}(\widehat{\theta_l}) \cdot g(\widehat{\theta_l}), \quad (8)$$

until a sufficiently accurate estimation is reached. This idea is summarized in the following algorithm.

Algorithm 1.
(A Maximum Likelihood scheme for the set intersection estimation problem)
The scheme gets as an input the sketches of the sets $\{x_A^k\}_{k=1}^m$ and $\{x_B^k\}_{k=1}^m$, where x_A^k and x_B^k are the maximal hash values of A and B respectively for the kth hash function, and returns an estimate of their set intersection cardinality.

1) *Estimate* $a_0 = \widehat{a}$, $b_0 = \widehat{b}$ *and* $\widehat{u}$ *using any cardinality estimation algorithm, such as [12].*
2) *Estimate the Jaccard similarity* $\widehat{\rho}$ *from the given sketches of* A *and* B.
3) *Find the maximum of the likelihood function* L *(Eq. (5)) as explained above; use* $n_0 = \widehat{\rho} \cdot \widehat{u}$ *as an initial value of* n *(see Scheme-2 in Section 3), and* a_0, b_0 *as an initial values of* a *and* b *respectively.*
4) *Return* $\widehat{n}$.

When we implemented Algorithm 1, we discovered that 3 Newton-Raphson iterations are enough for the algorithm to converge. Each Newton-Raphson step requires only a simple inversion of a 3x3 matrix and thus incurs only negligible computational effort. However, instead of reaching full convergence, we can use one of the "previous schemes" (namely, Scheme-1, Scheme-2 or Scheme-3 from Section 3) as initial estimation, and then run only a single Newton-Raphson iteration to improve the result. **Therefore, while our new scheme performs significantly better than Scheme-1, Scheme-2, and Scheme-3 (has smaller variance), it does not require a longer running time.**

4.3 The Optimal Variance of the New Estimator

The new estimator proposed above is based on Maximum Likelihood and thus it asymptotically achieves optimal variance [23] with respect to min/max-sketch algorithms[3]. Note that all main previous works, and in particular the three schemes described in Section 3, are based upon min/max-sketches. We now use the Cramer-Rao bound to compute this optimal variance.

The Cramer-Rao bound states that the inverse of the Fisher information matrix is a lower bound on the variance of any unbiased estimator [23]. The Fisher information matrix $\mathbb{F}_{i,j}$ is a way of measuring the amount of information that a random variable X carries about an unknown parameter θ upon which the probability of X depends. It is defined as:

$$\mathbb{F}_{i,j} = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log L\right]. \tag{9}$$

We now use the log-likelihood function (Eq. (5)) to derive this matrix for the set intersection estimation problem:

$$\mathbb{F}(a,b,n) = m \cdot \begin{pmatrix} \frac{\beta}{u}\cdot\frac{1}{a^2} + \frac{1}{u\cdot\alpha} & 0 & \frac{-1}{u\cdot\alpha} \\ 0 & \frac{\alpha}{u}\cdot\frac{1}{b^2} + \frac{1}{u\cdot\beta} & \frac{-1}{u\cdot\beta} \\ \frac{-1}{u\cdot\alpha} & \frac{-1}{u\cdot\beta} & \frac{1}{u\cdot n} + \frac{1}{u\cdot\beta} + \frac{1}{u\cdot\alpha} \end{pmatrix},$$

where each term is derived due to algebraic manipulations and derivatives of the log-likelihood function. Note that the expected values of the indicator variables $I_{1,k}, I_{2,k}$ and $I_{3,k}$ are required to derive the matrix. For $I_{1,k}$ we get:

$$\mathbb{E}[I_{1,k}] = \Pr\left(x_A^k = x_B^k\right) = \frac{n}{u} \quad , 1 \le k \le m\,.$$

[3]The optimality holds also with respect to bottom-k sketches, as they are statistically equivalent to min/max-sketches.

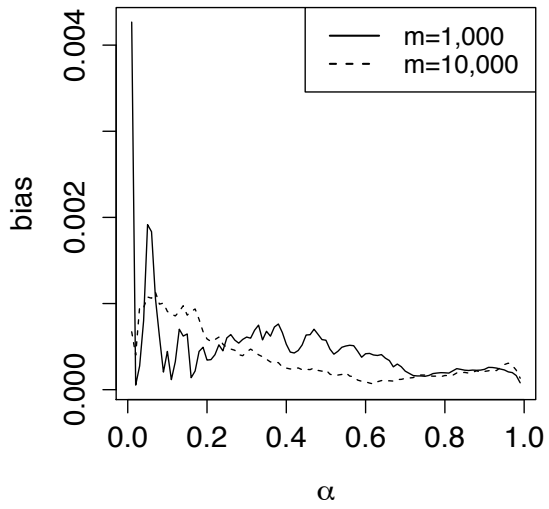


Figure 1: The bias of the new ML scheme for $f = 1$ and for different α values, for $m = 1,000$ and $m = 10,000$

The first equality is due to the definition of $I_{1,k}$, and the second is due to [3]. The following are obtained in the same way for every $1 \le k \le m$:

(1) $\mathbb{E}[I_{2,k}] = \Pr\left(x_A^k < x_B^k\right) = \frac{\beta}{u}$.
(2) $\mathbb{E}[I_{3,k}] = \Pr\left(x_A^k > x_B^k\right) = \frac{\alpha}{u}$.

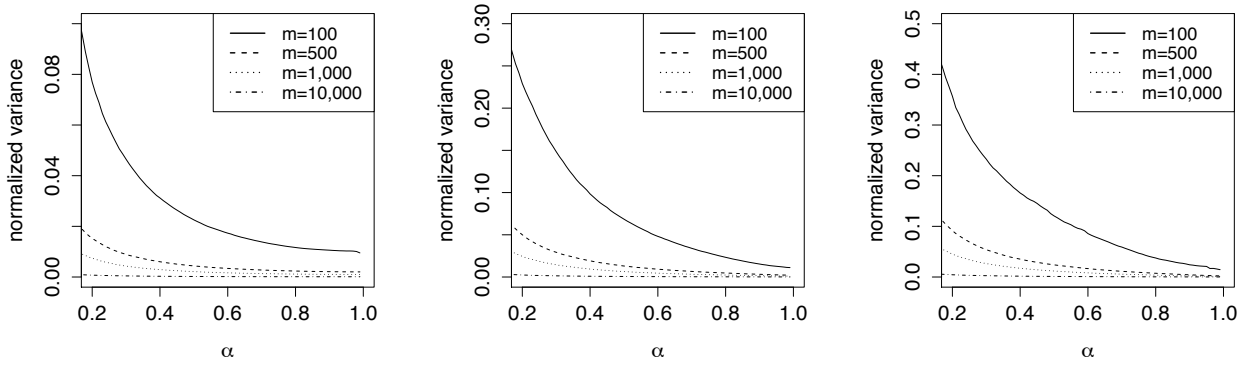
Let $|\widehat{A \cap B}|$ be an unbiased estimator for the set intersection estimation problem. Then, according to the Cramer-Rao bound, $\mathrm{Var}\left[|\widehat{A \cap B}|\right] \ge (\mathbb{F}^{-1})_{3,3}$, where $(\mathbb{F}^{-1})_{3,3}$ is the term in place $[3,3]$ in the inverse Fisher information matrix. Finally, from the computation of the term $(\mathbb{F}^{-1})_{3,3}$, we can obtain the following corollary:

Corollary 4.2.
$\mathrm{Var}\left[|\widehat{A \cap B}|\right] \ge (\mathbb{F}^{-1})_{3,3}$, *where,*

$$(\mathbb{F}^{-1})_{3,3} = \frac{n\cdot u}{m} \cdot \frac{(b^2+\alpha\beta)(a^2+\alpha\beta)}{\alpha\cdot n(a^2+\alpha\beta)+\beta\cdot n(b^2+\alpha\beta)+(a^2+\alpha\beta)(b^2+\alpha\beta)}.$$

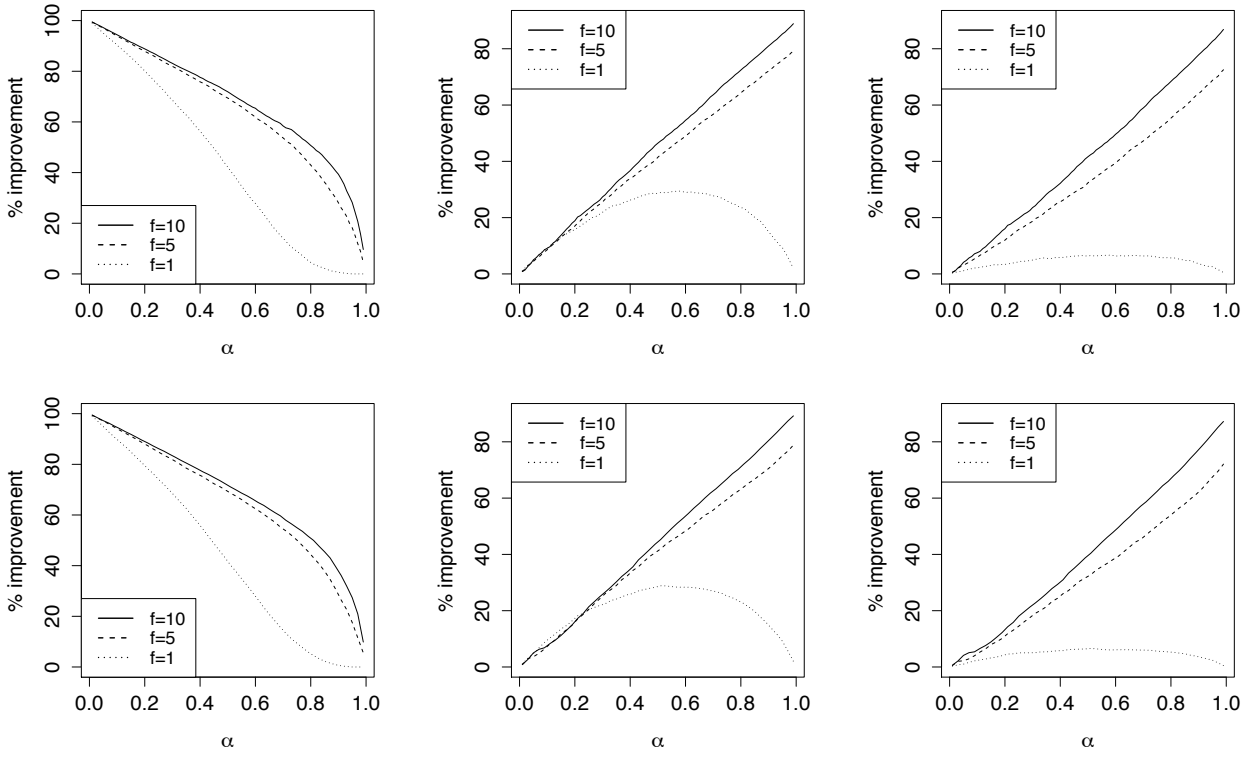
5 SIMULATION STUDY

In this section we examine the performance of our new ML estimator and show that it indeed outperforms the three known schemes when using the same memory. We simulate different data sets and estimate set intersection cardinality using the new scheme as well as the three previous schemes. We then measure the improvement (in percentages) of the new scheme over each previous scheme.



(a) $f = 1$ (b) $f = 5$ (c) $f = 10$

Figure 2: The normalized variance of the new ML scheme for different values of f, α and m



(a) Scheme-1 (b) Scheme-2 (c) Scheme-3

Figure 3: The percentage of variance improvement of the new ML scheme over each of the other schemes for different values of f and α; the upper graphs are for $m = 10,000$ and the lower graphs are for $m = 1,000$

In the simulation, we use the HyperLogLog algorithm [12] to estimate $|A|$, $|B|$ and $|A \cup B|$. We then estimate $\widehat{|A \cap B|}$ for each of the four schemes, for $f \in \{1, 5, 10\}$ and for $\alpha \in \{0, 0.01, 0.02, \ldots, 0.98, 0.99, 1\}$, where

(1) $|A| = a = 10^6$;
(2) $|B| = a \cdot f$, where $f > 0$;
(3) $|A \cap B| = a \cdot \alpha$, where $0 \leq \alpha \leq 1$.

We repeat the test for $10,000$ different sets. Thus, for each of the four schemes, and for each f and α values, we get a vector of $10,000$ different estimations. Then, for each f and α values, we compute the variance and bias of this vector, and view the result as the variance and bias of the estimator (for the specific f and α values). Each such computation is represented by one point in the graph. Let $v_{f,\alpha} = (\widehat{n}_1, \ldots, \widehat{n}_{10^4})$ be the vector of estimations for a specific scheme and for specific f and α values. Let $\mu = \frac{1}{10^4} \sum_{i=1}^{10^4} \widehat{n}_i$, be the mean of $v_{f,\alpha}$. The bias and variance of $v_{f,\alpha}$ are computed as follows:

$$\text{Bias}(v_{f,\alpha}) = \left| \frac{1}{n}(\mu - n) \right|$$

and

$$\text{Var}\left[v_{f,\alpha}\right] = \frac{1}{10^4} \sum_{i=1}^{10^4} (\widehat{n}_i - \mu)^2.$$

Figure 1 presents the bias of the ML estimator for $f = 1$, different α values, and two values of m: $m = 1,000$ and $m = 10,000$ (recall that m is the number of hash values used for the estimations of $\widehat{|A|}$, $\widehat{|B|}$ and the Jaccard similarity $\widehat{\rho}$). We can see that the bias is very small for all α and m values. We got very similar results for bigger f values as well.

Figure 2 presents the normalized variance ($\text{Var}\left[\frac{\widehat{n}}{n}\right]$) of our ML estimator for different f and α values, and for $m \in \{100, 500, 1000, 10000\}$. As expected, the normalized variance decreases as the number of hash values (m) increases, or as α increases. Overall, the normalized variance is very small for all values of α, f and m, indicating that the new scheme is very precise.

After showing that the new ML scheme indeed yields good results, we now compare its performance to that of Schemes 1-3. When comparing the statistical performance of two algorithms, it is common to look at their MSE (mean squared error) or RMSE, where $\text{MSE} = (\text{Bias}(\widehat{\theta}))^2 + \text{Var}\left[\widehat{\theta}\right]$, and $\text{RMSE} = \sqrt{(\text{Bias}(\widehat{\theta}))^2 + \text{Var}\left[\widehat{\theta}\right]}$. In our case, because all the estimators are unbiased, we compare only their variance. We define the "relative variance improvement" of the new ML scheme over each scheme as

$$\frac{\text{Var}\left[\widehat{\theta_i}\right] - \text{Var}\left[\widehat{\theta_{ML}}\right]}{\text{Var}\left[\widehat{\theta_i}\right]},$$

where θ_i is the estimator of the ith scheme, and θ_{ML} is the new ML estimator.

Figure 3 presents the simulation results for two values of m: $m = 10,000$ (upper graphs) and $m = 1,000$ (lower graphs). **We can see that the new ML estimator outperforms the three schemes for all values of α and f, and for both values of m.** This improvement varies between 100% to a few percent.

6 ANALYSIS OF THE THREE PREVIOUS SCHEMES

We have conducted a full analysis of the asymptotic statistical performance (bias and variance) of the three previous schemes. Due to space constraints, this analysis is not included in this short version of the paper, and the reader is referred to the full technical report [10]. For the sake of completeness, we give here the final results of the analysis.

Analysis of Scheme-1 (Theorem 8 in the full paper [10])

$\frac{\widehat{n_1}}{n} \to \mathcal{N}(1, V)$, where

$$V = \frac{1}{mn^2}(u^2 - a^2 - b^2) - \frac{2a \cdot b}{m \cdot u \cdot n} + \frac{2u \cdot (a^2(b^2 + \alpha\beta) + b^2(a^2 + \alpha\beta))}{m \cdot Z \cdot n},$$

In addition m is the number of storage units, and Z satisfies:

$$Z = \alpha \cdot n(a^2 + \alpha\beta) + \beta \cdot n(b^2 + \alpha\beta) + (a^2 + \alpha\beta)(b^2 + \alpha\beta).$$

Analysis of Scheme-2 (Theorem 9 in the full paper [10])

$$\frac{\widehat{n_2}}{n} \to \mathcal{N}\left(1, \frac{1}{m\rho}\right),$$

where m is the number of storage units.

Analysis of Scheme-3 (Theorem 11 in the full paper [10])

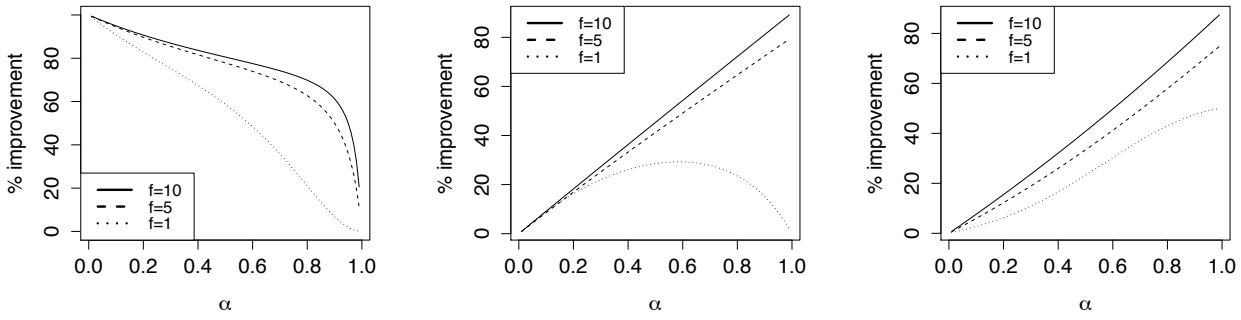
$$\frac{\widehat{n_3}}{n} \to \mathcal{N}\left(1, \frac{1}{m}(1 + \frac{2ab}{u(a+b)} + (\alpha + \beta)\frac{u^2}{n(a+b)^2})\right),$$

where m is the number of storage units.

To verify that the asymptotic variance of the new scheme is indeed better (lower) than the asymptotic variances of the three previous schemes, we compute the analyzed asymptotic variance of each of the four schemes according to Corollary 4.2 and the theorems above from the full paper [10], for $f \in \{1, 5, 10\}$ and for $\alpha \in \{0, 0.01, 0.02, \ldots, 0.98, 0.99, 1\}$; as in Section 5, $f = \frac{|B|}{|A|}$ and $\alpha = \frac{|A \cap B|}{|A|}$. Figure 4 shows the improvement (in percentages) of the analyzed asymptotic variance of the new scheme over each of the three previous schemes. The results are for $|A| = 10^6$ and $m = 100$, but note that the improvement is affected only by α and f and not by the value of m.

We can see that the analyzed asymptotic variance of the new ML estimator indeed outperforms the analyzed asymptotic variances of the three schemes for all values of α and f.

One can see the similarity between this figure, which presents the **expected** improvement of the new scheme over each of the three schemes according to their analysis, and Figure 3, which presents the **actual** improvement according to the simulations variance. In other words, the improvement of the new scheme over each scheme in the simulations in Section



(a) Improvement over Scheme-1 (b) Improvement over Scheme-2 (c) Improvement over Scheme-3

Figure 4: The percentage of variance improvement of the new ML scheme over each of the other schemes for different values of f and α

5 is very close to the improvement expected by our analysis of the four schemes.

7 CONCLUSION

In this paper we studied the problem of estimating the number of distinct elements in the set intersection of two streams. We computed the likelihood function of the problem and used it to present a new estimator, based on the ML method. We also found the optimal variance of any unbiased set intersection estimator, which is asymptotically achieved by our new ML scheme. We can conclude that our new scheme outperforms the three known schemes, significantly improves the variance (precision) of the estimator, and yields better results than the three previously known schemes.

REFERENCES

[1] J. Bauckmann, U. Leser, F. Naumann, and V. Tietz. Efficiently detecting inclusion dependencies. In *ICDE*, pages 1448–1450, 2007.
[2] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD Conference*, pages 199–210, 2007.
[3] A. Z. Broder. On the resemblance and containment of documents. In *IEEE Compression and Complexity of Sequences 1997*, pages 21–29.
[4] A. Z. Broder. Identifying and filtering near-duplicate documents. In *CPM*, pages 1–10, 2000.
[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.
[6] P. Chassaing and L. Gérin. Efficient estimation of the cardinality of large data sets. In *Proceedings of the 4th Colloquium on Mathematics and Computer Science*, volume AG of *Discrete Mathematics & Theoretical Computer Science Proceedings*, pages 419–422, 2006.
[7] P. Clifford and I. A. Cosma. A statistical analysis of probabilistic counting algorithms. *Scandinavian Journal of Statistics*, 2011.
[8] C. Clifton, E. Housman, and A. Rosenthal. Experience with a combined approach to attribute-matching across heterogeneous databases. In *DS-7*, pages 428–451, 1997.
[9] E. Cohen and H. Kaplan. Tighter estimation using bottom k sketches. *PVLDB*, 1(1):213–224, 2008.
[10] R. Cohen, L. Katzir, and A. Yehezkel. A minimal variance estimator for the cardinality of big data set intersection. *CoRR*, abs/1606.00996, 2016.
[11] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *SIGMOD Conference*, pages 240–251, 2002.
[12] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Analysis of Algorithms (AofA) 2007*. DMTCS.
[13] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31:182–209, Sep. 1985.
[14] É. Fusy and F. Giroire. Estimating the number of active flows in a data stream over a sliding window. In D. Panario and R. Sedgewick, editors, *ANALCO*, pages 223–231. SIAM, 2007.
[15] S. Ganguly, M. N. Garofalakis, R. Rastogi, and K. K. Sabnani. Streaming algorithms for robust, real-time detection of ddos attacks. In *ICDCS*, page 4. IEEE Computer Society, 2007.
[16] F. Giroire. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics*, 157:406–427, 2009.
[17] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.
[18] H. Köhler. Estimating set intersection using small samples. In *ACSC*, pages 71–78, 2010.
[19] J. Lumbroso. An optimal cardinality estimation algorithm based on order statistics and its full analysis. In *Analysis of Algorithms (AofA) 2010*. DMTCS.
[20] A. Metwally, D. Agrawal, and A. E. 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*, EDBT '08, pages 618–629.
[21] A. E. Monge. Matching algorithms within a duplicate detection system. *IEEE Data Eng. Bull.*, 23(4):14–20, 2000.
[22] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.
[23] J. Shao. *Mathematical Statistics*. Springer, 2003.
[24] D. Ting. 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, 2016*, pages 1195–1204.