Communication-Efficient Computation on Distributed Noisy Datasets

Qin Zhang*
Indiana University Bloomington
qzhangcs@indiana.edu

ABSTRACT

This paper gives a first attempt to answer the following general question: Given a set of machines connected by a point-to-point communication network, each having a *noisy* dataset, how can we perform communication-efficient statistical estimations on the union of these datasets? Here 'noisy' means that a real-world entity may appear in different forms in different datasets, but those variants should be considered as the same universe element when performing statistical estimations. We give a first set of communication-efficient solutions for statistical estimations on distributed noisy datasets, including algorithms for distinct elements, L_0-sampling, heavy hitters, frequency moments and empirical entropy.

1. INTRODUCTION

In many of today's applications, data is distributed in different machines/sites which are connected by a network. The sites need to answer queries defined on the union of their datasets. Different from the traditional centralized data processing where the primary goal is to minimize the number of cells/blocks probed in the RAM/disk, in the distributed setting we are mainly interested in minimizing the total bits of communication between sites and the total number of rounds of the computation, since they directly link to the network bandwidth and energy consumption, and typically dominate the total running time of the computation. These two measurements are captured by various distributed/parallel computational models, such as the BSP model [48], the $\mathcal{MRC}$ MapReduce [35], the generic MapReduce model [25] and the Massively Parallel model [39].

In this paper we consider statistical estimations in the distributed model, including computing distinct elements, heavy hitters, frequency moments and entropy, all of which are fundamental problems in data analytics. A natural way to communication-efficiently compute these statistics in the distributed model is to use linear sketches developed in the data stream literature, such as AMS-sketch [5] for frequency moments and Count-Min sketch [16] / Count-Sketch [14] for point queries. We can designate an arbitrary site as the *coordinator*, and every other site simply computes a linear sketch of its local data and sends it to the coordinator. The coordinator adds up all local linear sketches to obtain a sketch for the global data, and then extracts the answer from the global sketch. In this paper we consider a more challenging question:

> *What if data objects in different sites are* noisy*, that is, the same real-world entity may appear in different forms in different datasets?*

We list a few scenarios where noisy data is generated.

- People may upload the same video/image to YouTube/Flickr with different sizes, formats and compression ratios.
- Queries of the same meaning may be sent to Google using different keywords combinations.
- Customers may use different spellings of their shipping addresses in their online purchases from different merchants.

Note that these types of noise, if not treated properly, will cause incorrect results in data analytics. For example, if we do not treat 100 images of the same object but compressed in different rates as the same entity, then the distinct elements of this dataset changes from 1 to 100, and the entropy changes from 0 to $\log 100$, which is clearly unacceptable in practice.

Noisy data is universal and has caused significant obstruction to business practices [20, 22]. How to manage noisy datasets has been studied for several decades under different names, e.g., "entity resolution", "de-duplication", "reference reconciliation", "record linkage", etc. See surveys and book chapters [18, 21, 27, 37] for introductions. However, most of the proposed methods are designed for centralized computations, and focus on detecting items representing the same entity and unifying their representations, and/or outputting all distinct entities. The de-duplication procedure usually involves complicated inference models, such as relational approaches [6, 32] and collective approaches [12, 47]. In other words, those approaches target a comprehensive de-duplication, and thus have a time/space complexity at least linear (often much larger) in terms of the input size. While in this paper we consider the "big data" setting, where we assume it is relatively easy to determine whether two given items are duplicates or not (by a pairwise comparison using some distance measure), the sizes of the datasets and the number of duplicates are huge. We are interested in algorithms for statistical queries with *sublinear* communication cost and *constant or logarithmic* rounds at a modest cost of accuracy, and thus we cannot afford a comprehensive de-duplication. Therefore our setting is very different from the traditional ones for entity resolution.

This work is largely motivated by noticing that the aforementioned linear sketching approach can not be used directly for noisy

*This project was funded by IU's Office of the Vice Provost for Research through the Faculty Research Support Program.

datasets, simply because frequencies of items representing the same entity may *not* be mapped into the same bucket in the sketch and added together if sites do not communicate with each other before performing linear sketching individually. On the other hand, as mentioned above, converting noisy datasets to noise-free ones before sketching is communication expensive. Thus new approach is needed to obtain communication efficient algorithms for distributed noisy datasets.

Robust Statistics on Distributed Noisy Datasets. Now we formally define our model. We have k sites, each holding a multiset of items S_i. Let multiset $S = \bigcup_{i\in[k]} S_i$. The premise is that items in S can be partitioned into a set of groups (multisets) $\mathcal{G} = \{G_1, \ldots, G_n\}$ (for an unknown n) such that items in the same group represent the same real-world entity. The k sites would like to jointly compute some statistical function f defined on S by treating items from the same group as the same item. For example, the distinct elements function is defined to be $F_0(S) = |\mathcal{G}| = n$. We always allow a $(1+\epsilon)$-approximation since for exact computation, in the worst case, there is often no better way than shipping all items to one site (for many statistical problems, this holds even in the noise-free case, see [50]). The precise meaning of the $(1+\epsilon)$-approximation depends on specific problems.

We assume that each site is quipped with a *comparison metric* which, given two items, can determine whether they belong to the same group. We also assume transitivity, that is, if u, v represent the same entity (write $u \sim v$) and $v \sim w$, then $u \sim w$. This assumption guarantees a unique partition of $\mathcal{G} = \{G_1, \ldots, G_n\}$, and was used in literature [26, 44].

In fact, using a local comparison metric we can interpret a "real-world entity" in a broader sense: it can be a topic, a class, etc. In other words, we can perform statistical analysis directly on groups for arbitrary similarity-based clusterings of the dataset. For example, our algorithms can also be used for queries such as how many search topics over the union of the k datasets each of which consists of a bag of search keywords.

Justifying the Model and Assumptions. Since the model studied in this paper is new, we would like to put a few remarks trying to answer the questions readers may have, including why do we use a local comparison model with a transitivity assumption? Whether there exist some "magic" hash functions that can map all items in the same group to the same entity and make our problems easy? Can the parties resolve locally their own entity-resolution problem by considering a shared vocabulary/ontology? The discussions on these questions can be found in Appendix A.

The Coordinator Model. For the convenience of presenting our algorithms, we introduce an extra party called the *coordinator* whose input is an empty set (we can also designate an arbitrary site as the coordinator). We divide the whole computation into *rounds*. In each round the coordinator sends a message of arbitrary length to each site, and then each site who has been contacted may send a message of arbitrary length to the coordinator. A round is completed when the coordinator has received a message from each site from whom it is expecting to get a response. We call this model the coordinator model. Our goal is to minimize the total bits of communication between all parties and the total number of rounds of the computation. Note that the underlying communication topology of the coordinator model is in fact a clique, up to a factor of 2 (the k sites can use the coordinator as the router), thus the coordinator model is essentially equivalent to the case when we allow sites to have direct point-to-point communication with each other.

The coordinator model has attracted a lot of attentions in recent years [1, 28, 46, 50]. In the high level, it is similar to the congested clique model [19, 40, 41, 45] and the k-machine model [36]. The main difference is that in the coordinator model there is no bandwidth limit on each coordinator-site communication channel. However, in our algorithms the amount of communication at each communication channel are similar since sites' positions are *symmetric*. Moreover, in all of our algorithms the total bits of communication is sublinear (low polynomials in k and $1/\epsilon$, see Table 1), thus the number of rounds will be sublinear even when in each round only 1 bit can be communicated through each coordinator-site channel.

Our Contributions. This work aims to initiate the study of designing communication-efficient algorithms for distributed noisy data objects *without* a comprehensive but still resource-consuming data de-duplication step, which is critical to queries in scenarios where data is massive, distributed and constantly evolving. We will focus on statistical estimations on the datasets. In particular, we consider distinct elements, frequency moments, L_0-sampling, heavy hitters and empirical entropy, all of which are basic measurements for understanding the distribution of a dataset, and are well-studied (on noise-free datasets) in the streaming model (e.g. [5, 9, 13, 23, 33]), the coordinator model (e.g., using linear sketches mentioned before, or mergeable summaries [1]) and the distributed monitoring model[1] [8, 17, 29, 49].

We first give the precise definitions of these problems, and then state our results. Denote $[t] = \{1, 2, \ldots, t\}$. Let $[N]$ be the item universe. Let $m = |S|$ be the total number of items in the union of the k datasets, and let n be the number of groups in S (i.e., $|\mathcal{G}|$).

1. *p-th frequency moment* of S: $F_p(S) = \sum_{i\in[n]} |G_i|^p$. When $p = 0$, F_0 simply counts the number of groups (distinct universe elements) of S (i.e., $|\mathcal{G}|$). In general, F_p can be used to measure how skew a dataset is. By allowing a $(1+\epsilon)$-approximation we mean that the algorithm can return any value in $[(1-\epsilon)F_p(S), (1+\epsilon)F_p(S)]$.

2. L_0*-sampling* on S: return a group G_i (or an arbitrary item in G_i) uniformly at random from $\mathcal{G}$. Besides being used to perform *duplicate-free* sampling, L_0-sampling is the key tool for designing graph sketches [2, 3].

3. ϕ*-heavy-hitter* ($0 < \phi \leq 1$) of S: return those "heavy" groups whose frequency is larger than ϕm. If we allow an ϵ-approximation ($0 < \epsilon \leq \phi$), then we can return a set $\mathcal{G}' \subseteq \mathcal{G}$ containing all groups G (or an arbitrary item from each group G) such that $|G| \geq \phi m$, and no group G such that $|G| \leq (\phi - \epsilon)m$. Decisions for groups G with $(\phi-\epsilon)m \leq |G| \leq \phi m$ can be made arbitrarily. We call the relaxed version (ϕ, ϵ)*-heavy-hitter* of S.

4. *Empirical entropy*: $\text{Entropy}(S) = \sum_{i\in[n]} \frac{|G_i|}{m} \log \frac{m}{|G_i|}$. Empirical entropy is very effective to detect subtle changes of the data distribution.

We summarize our results in Table 1. For simplicity, we use $\tilde{O}(f)$ to denote $f \cdot \text{poly}\log(f \cdot kmN)$, and for technical convenience, we assume that $m \geq n \gg \{k, 1/\epsilon\} \gg \log(mN)$. We usually think k is larger than $1/\epsilon$ since one of the main goals of distributed/parallel computation is to scale to a large number of sites, while ϵ in many cases can be thought as a constant. In this sense

[1]The distributed monitoring model can be seen as a continuous version of the coordinator model, where data is streaming in at each of the k sites and we want to compute the function at any time step (see the formal definition by Cormode et al. [17]). Obviously, any communication upper bound for distributed monitoring also holds for the coordinator model.

	noisy datasets		noise-free datasets	
	(comm.) bits UB	rounds UB	bits UB	bits LB
F_0	$\tilde{O}(\min\{k/\epsilon^3, k^2/\epsilon^2\})$	$\tilde{O}(1)$	$\tilde{O}(k/\epsilon^2)$ * [17]	$\Omega(k/\epsilon^2)$ [49, 51]
L_0-sampling	$\tilde{O}(k)$	$\tilde{O}(1)$	$\tilde{O}(k)$	$\Omega(k)$
F_p $(p \geq 1)$	$\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$	$O(1)$	$\tilde{O}((k^{p-1}/\text{poly}(\epsilon))$ * [49]	$\Omega(k^{p-1}/\epsilon^2)$ [49]
(ϕ, ϵ)-HH	$\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$	1	$\tilde{O}(\min\{\frac{\sqrt{k}}{\epsilon}, \frac{1}{\epsilon^2}\})$ * [29]	$\Omega(\min\{\frac{\sqrt{k}}{\epsilon}, \frac{1}{\epsilon^2}\})$ [49]
Entropy	$\tilde{O}(k/\epsilon^2)$	$O(1)$	$\tilde{O}(k/\epsilon^2)$ * [15]	$\Omega(k/\epsilon^2)$ ** [49]

Table 1: Our results. HH denotes heavy-hitter. UB and LB denote upper bound and lower bound respectively. We compare our communication costs with the upper and lower bounds of that for noise-free datasets. *These upper bounds hold even in the (continuous) distributed monitoring model. **This lower bound requires item deletions.

k/ϵ^3 is better than k^2/ϵ^2 for F_0. Moreover, in our algorithm for L_0-sampling we use an algorithm for computing F_0 with ϵ setting to be a constant as a subroutine, thus using the first bound $O(k/\epsilon^3)$ for F_0 will help to save a factor of k for L_0-sampling, which is significant.

We also compare our results with the corresponding bounds for the noise-free datasets in the coordinator model. Somewhat surprisingly, our upper bounds for the noisy datasets match or almost match the corresponding lower bounds for the noise-free datasets.

A Brief Technical Overview. As mentioned, some popular techniques such as hashing and linear sketching in literature cannot be used for handling noisy datasets. We thus need new approaches.

Our main results are algorithms for F_0 and an algorithm for L_0-sampling, presented in Section 3 and Section 4 respectively. For F_0 we give two algorithms. The first algorithm (Section 3.1) is an adaptation of the BJKST algorithm [9] by explicitly resolving item duplications in a final clean up step. This gives an $\tilde{O}(k^2/\epsilon^2)$ bound on communication and $O(1)$ rounds.

Our main technical contribution is the second algorithm for F_0 presented in Section 3.2.2, which achieves an $\tilde{O}(k/\epsilon^3)$ bound on communication. We proceed in two steps. We first observe a simple sampling approach using the fact that the variance can be bounded if we can reduce the maximum duplication of items to k by an initial local de-duplication step. We next further reduce the variance of the second algorithm (Section 3.2.1) by partitioning items to classes based on their duplication factors. However, it is impossible to perform a complete classification in the distributed setting without a signification amount of communication, since we have to spend some bits on each of the items. To bypass this difficulty, we first perform a local hierarchical sampling, and then use a rejection sampling process to classify items *on the fly*. In this way we only need to classify those *sampled* items, but how to bound the number of sampled items involves quite some subtleties.

For those who are familiar with sampling algorithms, we would like to comment that our distributed hierarchical sampling algorithm is very different from the now standard sub-sampling methods by Indyk and Woodruff [31] in the streaming model, and the hierarchical sampling algorithm by Gibbons and Tirthapura [24] in the distributed streaming model. The main difference is that in our setting, hierarchical sampling at each site is performed *independently*, and items reference to the same entity may be sampled at different levels at different sites because there does *not* exist a global magic hash function to assign levels to items of the same group consistently. This difference makes our hierarchical sampling algorithm and the corresponding analysis more complicated than that in [24].

In our L_0-sampling algorithm, we make use of a random shuffling step plus a synchronization step to obtain the optimal communication cost (up to some log factors).

The algorithms for heavy hitters, frequency moments and entropy, presented in Section 5, are straightforward adaptations of existing algorithms for the noise-free data setting. However, one needs to pick the right algorithms (for noise-free datasets) to extend. We note that most algorithms for F_p in the literature (e.g., [31]), cannot be used for noisy datasets.

2. PRELIMINARIES

We write $u \sim v$ if u and v belong to a same group G, and $u \not\sim v$ otherwise. Given $u \in S$, let $G_{(u)}$ be the group containing u. Let $a \in_R A$ denote a sample a chosen uniformly at random from A.

We need the following versions of the Chernoff bound.

Lemma 1 (Standard Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables such that $\mathbf{Pr}[X_i = 1] = p_i$. Let $X = \sum_{i\in[n]} X_i$. Let $\mu = \mathbf{E}[X]$. It holds that $\mathbf{Pr}[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}$ and $\mathbf{Pr}[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2}$ for any $\delta \in (0, 1)$.*

Lemma 2 *Let $X_1, \ldots, X_n$ be independent scalar random variables with $|X_i| \leq M$ almost surely, with mean μ_i and variance σ_i^2. Let $X = \sum_{i\in[n]} X_i$. Then for any $\mu > 0$,*

$$\mathbf{Pr}(|X - \mu| \geq \lambda\sigma) \leq C \max\{e^{-c\lambda^2}, e^{-c\lambda\sigma/M}\}$$

for some absolute constants $C, c > 0$, where $\mu = \sum_{i=1}^n \mu_i$ and $\sigma^2 = \sum_{i=1}^n \sigma_i^2$.

Lemma 3 *Let $Y_1, \ldots, Y_n$ be n independent random variables such that $Y_i \in [0, T]$ for some $T > 0$. Let $\mu = \mathbf{E}[\sum_i Y_i]$. Then for any $a > 0$, we have $\mathbf{Pr}\left[\sum_{i\in[n]} Y_i > a\right] \leq e^{-(a-2\mu)/T}$.*

We also need the following observation.

Observation 1 *The coordinator can sample s items from S uniformly at random with replacement using $\tilde{O}(k+s)$ bits of communication and $O(1)$ rounds.*

PROOF. First each site i sends $|S_i|$ to the coordinator and the coordinator computes $|S| = \sum_{i\in[k]} |S_i|$. Next, the coordinator samples s sites from the k sites with replacement such that the probability that site i is sampled each time is $|S_i| / |S|$, and then each sampled site i samples an item uniformly at random from S_i and sends to the coordinator. □

3. THE DISTINCT ELEMENTS PROBLEM

In this section we give algorithms for distinct elements (F_0). W.l.o.g., we assume that for any S_i, for any $u, v \in S_i$, we have

Algorithm 1: Estimating F_0 for Distributed Noisy Datasets by Extending BJKST

1 the coordinator picks a random hash function $h : [N] \to [N]$ from a 2-universal family, and sends it to each of the k sites;
2 each site i individually runs the BJKST algorithm with $\lambda = ck$ for a large enough constant c, and sends z_i and the set B_i to the coordinator;
3 the coordinator computes $z = \max\{z_i \mid i \in [k]\}$;
4 for each $i \in [k]$, the coordinator removes all pairs $(u, \text{zero}(h(u)))$ in B_i with $\text{zero}(h(u)) < z$, getting set $B_i' \subseteq B_i$;
5 (*synchronization*) the coordinator checks for each $i \in [k]$, for each $u \in B_i'$, if there exists a $j < i$ such that $v \in S_j$ and $u \sim v$, by communicating u with sites $1, \ldots, i-1$ in order. If yes, coordinator deletes u from B_i'. Let $B_1'', \ldots, B_k''$ be the sets of $B_1', \ldots, B_k'$ after the synchronization;
6 the coordinator computes set $B'' = \bigcup_{i \in [k]} B_i''$, and outputs $|B''| \, 2^z$.

$u \not\sim v$, that is, we only keep one item in the same group for each S_i. We can assume this because removing local duplicates will not affect the value of F_0 (or performing L_0-sampling in Section 4), and each site can remove local duplicates without any communication.

3.1 Warm Up: Extending BJKST

We first briefly sketch the BJKST algorithm for the noise-free data setting (slightly modified for our purpose). We choose a random hashing function $h : [N] \to [N]$ from a 2-universal hash family. For each item $u \in S$, we apply the hash function h on u and compute $\text{zeros}(h(u))$, which is the number of tailing zeros in the binary representation of $h(u)$. At the end we compute the largest number z such that there are at least λ/ϵ^2 (for a parameter λ specified later) items with $\text{zeros}(h(u)) \geq z$. We call z the *threshold* of the set S. We also maintain a set of pairs $B = \{(u, \text{zeros}(h(u))) \mid \text{zeros}(h(u)) \geq z\}$ during the run of the algorithm.

Lemma 4 ([9]) *For a large enough constant λ, the quantity $|B| \, 2^z$ is a $(1+\epsilon)$-approximation of the distinct elements of the set S in the noise-free setting with probability at least 0.99.*

Remark 1 The BJKST algorithm [9] is originally designed to work in the streaming model, thus we want to choose the *largest* z which will guarantee that $|\{u \in S \mid \text{zeros}(h(u)) \geq z\}| \in [\lambda/\epsilon^2, 4\lambda/\epsilon^2]$ with high probability, thus also the space usage. However, picking any z satisfying $|\{u \in S \mid \text{zeros}(h(u)) \geq z\}| \geq \lambda/\epsilon^2$ for a large enough constant λ is sufficient to make $|B| \, 2^z$ a $(1+\epsilon)$-approximation of $F_0(S)$ with probability at least 0.99. This hashing idea can be traced back to Flajolet-Martin [23] and Alon et al. [5] who gave constant approximations.

Our algorithm for noisy datasets is presented in Algorithm 1, where $c, z_i, B_i, B_i', B_i'', B''$ are defined. Line 1 to 4 is basically a run of the BJKST in the distributed setting, but we need to sample $\Theta(k/\epsilon^2)$ items even after the item removals at Line 4. The reason is that at Line 5 we need to perform a synchronization step to make sure that only one item in each group (the first one counting from site 1 to site k) is considered, so as to make the decisions (i.e., sampled or not) for items in a single group consistent. In other words, we are sampling groups, not individual items.

Algorithm 2: Estimating F_0 for Distributed Noisy Datasets by Sampling

1 the coordinator computes $m = \sum_{i \in [k]} |S_i|$ by contacting each site;
2 **for** $i = 1, \ldots, \eta_k$ **do**
3 the coordinator samples a random item $u_i \in S$ using Observation 1;
4 the coordinator computes $\left|G_{(u_i)}\right|$ by contacting k sites, and sets $X_i = 1/\left|G_{(u_i)}\right|$;
5 the coordinator outputs $\left(\frac{1}{\eta_k} \sum_{i \in [\eta_k]} X_i\right) m$.

Correctness. Let $B' = \bigcup_{i \in [k]} B_i'$. First, it is easy to see that $|B'| \geq ck/\epsilon^2$, since there is at least one $i \in [k]$ such that $z_i = z$, thus $B_i' = B_i$ and $|B_i'| \geq ck/\epsilon^2$. The synchronization step guarantees that only one item (the one appears in the site with the smallest index) in each group is considered in the "global" BJKST algorithm. Since $|B'| \geq ck/\epsilon^2$, and each group only has at most one item in each site (see the discussion at the beginning of Section 3), we have $|B''| \geq |B'|/k \geq c/\epsilon^2$. Thus Lemma 4 and Remark 1 give the correctness.

Complexities. Lines $1, 3$ cost $\tilde{O}(k)$ bits of communication. Lines $2, 4, 5$ cost $\tilde{O}(k^2/\epsilon^2)$ bits of communication. Line 6 is entirely local, and can be done by the coordinator without any communication. It is easy to see that this algorithm can be implemented in $O(1)$ rounds.

Theorem 1 *Algorithm 1 computes a $(1+\epsilon)$-approximation of $F_0(S)$ correctly with probability at least 0.99 in the distributed noisy data setting, using $\tilde{O}(k^2/\epsilon^2)$ bits of communication and $O(1)$ rounds.*

3.2 An Improved Algorithm (for Large k)

In this section we give an improved algorithm for robust F_0 when $k = \omega(1/\epsilon)$.

3.2.1 A Sampling Algorithm

We first introduce a simple sampling algorithm, which will be used in our improved algorithm as a subroutine. The sampling algorithm is presented in Algorithm 2, where u_i, X_i's are defined. Let $\eta_q = c_\eta \cdot q/\epsilon^2 \cdot \log(1/\delta)$ for a sufficiently large constant c_η.

Correctness. We will show that $\left(\frac{1}{\eta_k} \sum_{i \in [\eta_k]} X_i\right) m$ is a $(1+\epsilon)$-approximation of n with probability at least $1 - \delta$.

Let $\rho = n/m$. We have $1/k \leq \rho \leq 1$ since we can assume that each group only has at most one item in each site. Our goal is to show that $\left(\frac{1}{\eta_k} \sum_{i \in [\eta_k]} X_i\right)$ is a $(1+\epsilon)$-approximation of ρ, and consequently ρm will be a $(1+\epsilon)$-approximation of $n = F_0(S)$.

For each $i \in [\eta_k]$, we have

$$\begin{aligned} \mu_i = \mathbf{E}[X_i] &= \sum_{j \in [n]} \left(\mathbf{Pr}[u_i \in G_j] \cdot \frac{1}{|G_j|}\right) \\ &= \sum_{j \in [n]} \left(\frac{|G_j|}{m} \cdot \frac{1}{|G_j|}\right) \\ &= \frac{n}{m} = \rho. \end{aligned}$$

$$\begin{aligned}\sigma_i^2 = \mathbf{Var}[X_i] &= \mathbf{E}[X_i^2] - (\mathbf{E}[X_i])^2 \\ &= \sum_{j\in[n]} \left(\mathbf{Pr}[u_i \in G_j] \cdot \frac{1}{|G_j|^2}\right) - \left(\frac{n}{m}\right)^2 \\ &\le \frac{n}{m} - \left(\frac{n}{m}\right)^2 \qquad (|G_j| \ge 1) \\ &\le \rho.\end{aligned}$$

Let $X = \sum_{i\in[\eta_k]} X_i$. Then $\mu = \mathbf{E}[X] = \sum_{i\in[\eta_k]} \mu_i = \eta_k\rho$, and $\sigma^2 = \mathbf{Var}[X] = \sum_{i\in[\eta_k]} \sigma_i^2 \le \eta_k\rho$. Setting $\lambda = \epsilon\mu/\sigma$ in Lemma 2, noting that $X_i \le 1$ for all $i \in [\eta_k]$, by Lemma 2 we have

$$\begin{aligned}\mathbf{Pr}[|X-\mu| \ge \epsilon\mu] &\le C\max\{e^{-c(\epsilon\mu/\sigma)^2}, e^{-c(\epsilon\mu/\sigma)\sigma}\} \\ &\le C\max\{e^{-c\epsilon^2(\eta_k\rho)^2/(\eta_k\rho)}, e^{-c\epsilon(\eta_k\rho)}\} \\ &\le C\cdot e^{-c\epsilon^2/k\cdot\eta_k},\end{aligned}$$

which is at most δ if $\eta_k = c_\eta \cdot k/\epsilon^2 \cdot \log(1/\delta)$ for a sufficient large constant c_η. Therefore $\frac{1}{\eta_k}X = \left(\frac{1}{\eta_k}\sum_{i\in[\eta_k]} X_i\right)$ is a $(1+\epsilon)$-approximation of ρ with probability at least $1-\delta$.

Complexities. Set $\delta = 0.01$. Line 1 costs $\tilde{O}(k)$ bits of communication. Both Line 3 and 4 cost $\tilde{O}(k)$ bits of communication. Thus the total communication cost is $\tilde{O}(\eta_k \cdot k) = \tilde{O}(k^2/\epsilon^2)$. The number of rounds is $O(1)$ since we can run the algorithm for each sample in parallel.

Theorem 2 *Algorithm 2 computes a $(1+\epsilon)$-approximation of $F_0(S)$ correctly with probability at least 0.99 in the distributed noisy data setting, using $\tilde{O}(k^2/\epsilon^2)$ bits of communication and $O(1)$ rounds.*

Note that this sampling algorithm achieves the same complexity as Algorithm 1. We would like to include both since the idea in Algorithm 1 will be shared by the algorithm for L_0-sampling in Section 4, and this sampling algorithm will be used as a subroutine in our improved algorithm in the next subsection.

3.2.2 The Improved Algorithm

We now present our improved algorithm (when $k = \omega(1/\epsilon)$) for estimating robust F_0. The main idea is to reduce the variance of each X_i in Algorithm 2. Imagine that in the special case when the frequency of each group is either 1 or 2, the variance of each X_i in Algorithm 2 will be reduced by a factor of $\Theta(k)$ (in the worst case). Thus if we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_j = \{G \in \mathcal{G} \mid |G| \in (2^{j-1}, 2^j]\}$, and apply Algorithm 2 on each class individually, then we can shave a factor of k in the number of X_i needed (i.e., reduce the number of samples from η_k to η_2), thus also the communication cost. However, we cannot afford to partition the groups into classes in the distributed setting, because to do so we basically need to estimate the cardinality of each group up to a factor of 2, which needs $\Omega(F_0)$ bits communication.

In our new approach we do the following: Each site independently subsamples its items with probability $p_\ell = 1/2^\ell$ for $\ell = 0, 1, \ldots, L$ ($L = \log k$), which we call the *sample levels*. This naturally partitions all items (*not* groups) in S into a hierarchy of classes. As mentioned in the introduction, due to the lack of a global magic hash function, items in the same group may be sampled into different levels in different sites, which is very different from previous distributed/streaming sampling algorithms [24, 31], and is one of the major difficulties in our algorithm design and analysis.

More precisely, site i sets $V_i^0 = S_i$, and then for $\ell = 1, \ldots, L$, it constructs V_i^ℓ by subsampling each item in $V_i^{\ell-1}$ with probability $1/2$. Finally, site i sets $W_i^L = V_i^L$, and for $\ell = L-1, \ldots, 1, 0$, it sets $W_i^\ell = V_i^\ell \backslash V_i^{\ell+1}$. Let multiset $W^\ell = \bigcup_{i\in[k]} W_i^\ell$. Note that $\{W^0, \ldots, W^L\}$ is a partition of S. For a group $G \in \mathcal{G}$, let $G^\ell = G \cap W^\ell$ be the multiset of items in G whose maximum sample levels are ℓ.

The first natural idea is to estimate $F_0(S)$ as $\sum_{i=0}^L F_0(W^i)$, but there are two issues: First, we will have the problem of double-counting, that is, we may have $u \in W^\ell$ and $v \in W^{\ell'}$ ($\ell' \ne \ell$) such that $u \sim v$. In other words, two items in the same group may belong to two different W^ℓ's, and consequently this group will be counted at least twice. Second, for a level ℓ and an item $u \in W^\ell$, it may still be the case that $|G_{(u)}^\ell|$ is large, and then if we run Algorithm 2, the variance of the estimator X_i will again be high. To handle these two issues, we define $\tilde{W}^\ell \subseteq W^\ell$ for each $\ell = 0, 1, \ldots, L$ as follows.

Definition 1 Let $\tilde{W}^\ell$ be the multiset containing all items u satisfying the following.

1. $u \in W^\ell$.
2. There does not exists $v \in W^{\ell'}$ such that $u \sim v$ and $\ell' > \ell$. In other words, $\ell = \max\{\ell' \mid |G_{(u)}^{\ell'}| > 0\}$.
3. $|G_{(u)}^\ell| \le \tau$, where $\tau = 16\log m$.

The second constraint is used to avoid double-counting: each group G will only be counted at most once at the level $\max\{\ell \mid \left|G^\ell\right| > 0\}$. The third constraint is used to force the frequency of each group at each level to be no more than τ, for the purpose of reducing the variance when running Algorithm 2. However, by doing this it is possible that some groups are not counted at any level. Denote Q to be the set of such groups, then

$$F_0(S) = \sum_{\ell=0}^{L} F_0(\tilde{W}^\ell) + |Q|\,.$$

The following lemma shows that $|Q| = 0$ with high probability.

Lemma 5 *Let Q consist of all groups $G \in \mathcal{G}$ such that there exists an $\ell \in \{0, 1, \ldots, L\}$ with $\left|G^\ell\right| > \tau$, and there does not exist a $u \in G$ such that $u \in W^{\ell'}$ for an $\ell' > \ell$. We have that $|Q| = 0$ with probability at least $1 - 1/m$.*

PROOF. First, at level $\ell = L$, for each group G, by Lemma 3, noting that $\mu \le k \cdot 1/k = 1$, we have,

$$\mathbf{Pr}[|G^L| > \tau] \le e^{-(\tau-2)} \le 1/m^3.$$

By a union bound, with error probability at most $\delta_L = 1/m^3 \cdot n \le 1/m^2$, $\left|G^L\right| \le \tau$ holds for each group $G \in \mathcal{G}$.

We next consider any fixed level $\ell \in \{0, 1, \ldots, L-1\}$. For each group G, if $|G^\ell| > \tau$, then by a Chernoff bound we have

$$\begin{aligned}\mathbf{Pr}[|G^{\ell+1}| = 0] &\le \mathbf{Pr}\left[|G^{\ell+1}| < (1-99/100)\cdot|G^\ell|/2\right] \\ &\le e^{-\frac{(99/100)^2\cdot\left|G^\ell\right|/2}{2}} \le e^{-\tau/5} \le 1/m^3.\end{aligned}$$

By a union bound, with error probability at most $\delta_{<L} = 1/m^3 \cdot L \cdot n$, for any group $G \in \mathcal{G}$ and any level $\ell \in \{0, 1, \ldots, L-1\}$, if $|G^\ell| \ge \tau$, then $|G^{\ell+1}| \ge 1$, which means that G should be counted in $\tilde{W}^{\ell'}$ for some $\ell' > \ell$.

Combining the two cases, the probability that $|Q| = 0$ is at least $1 - \delta_L - \delta_{<L} \ge 1 - 1/m^2 - 1/m^3 \cdot L \cdot n \ge 1 - 1/m$. □

We say a level ℓ is *contributing* if $F_0(\tilde{W}^\ell) \geq (\epsilon/L) \cdot F_0(S)$, otherwise non-contributing. Then,

$$\begin{aligned} F_0(S) &\geq \sum_{\text{contributing } \ell} F_0(\tilde{W}^\ell) \quad (1) \\ &= F_0(S) - |Q| - \sum_{\text{non-contributing } \ell} F_0(\tilde{W}^\ell) \\ &\geq (1 - \epsilon/L \cdot L) F_0(S) \text{ (w. pr. } 1 - 1/m \text{ by Lemma 5)} \\ &= (1-\epsilon) F_0(S). \quad (2) \end{aligned}$$

Therefore to get a $(1 + O(\epsilon))$-approximation of $F_0(S)$, it suffices to estimate $F_0(\tilde{W}^\ell)$ for each contributing ℓ up to a $(1+\epsilon)$-approximation.

The difficulty of estimating $F_0(\tilde{W}^\ell)$ is that the k sites cannot compute $\tilde{W}^\ell$ exactly or even approximately without spending $\Omega(k)$ bits of communication to check if an item u is in $\tilde{W}^\ell$, which is communication prohibitive. We therefore have to check whether $u \in \tilde{W}^\ell$ when running Algorithm 2 on W^ℓ (from which sites can sample items easily, since site i knows W_i^ℓ exactly), and then reject those samples in $W^\ell \backslash \tilde{W}^\ell$.

Our algorithm is presented in Algorithm 3. We will show that for a contributing level ℓ, with high probability Algorithm 3 will reach Line 22, thus obtaining a $(1+\epsilon)$-approximation of $F_0(\tilde{W}^\ell)$.

One may observe that items from groups with large cardinalities will be sampled with a higher probability at Line 5, but those items are more likely in $W^\ell \backslash \tilde{W}^\ell$, thus we may waste communication on items that will be rejected eventually. Fortunately, we can show in the following lemma that such items can be rejected quickly, thus will not affect the efficiency of the rejection-sampling (to get a sample from $\tilde{W}^\ell$).

Lemma 6 *For a sample $u \in U$ at level ℓ, with probability at least $1 - 1/m^4$, we will contact at most $2\tau k/|G_{(u)}^\ell|$ (random) sites at Line 9 in Algorithm 3 before existing the while loop.*

PROOF. If $|G_{(u)}^\ell| \leq \tau$, then $2\tau k/|G_{(u)}^\ell| \geq k$. We thus only need to prove that the number of sites contacted at Line 9 is bounded by $2\tau k/|G_{(u)}^\ell|$ when $|G_{(u)}^\ell| > \tau$.

Let $\gamma = 2\tau k/|G_{(u)}^\ell|$. Let i_j $(j = 1, \ldots, \gamma)$ be the site sampled in the j-th trial (without replacement). Let $Y_j = 1$ if there exists a $v \in W_{i_j}^\ell$ such that $u \sim v$, and $Y_j = 0$ otherwise. Let $Y = \sum_{j \in [\gamma]} Y_j$. Let $\mu = |G_{(u)}^\ell|/k$. By a Chernoff bound (Chernoff bound also holds for sample without replacement, cf. [10]),

$$\mathbf{Pr}[Y \leq \tau] = \mathbf{Pr}[Y \leq \gamma\mu/2]] \leq e^{-\frac{(1/2)^2 \gamma\mu}{2}} = e^{-\tau/4} = 1/m^4.$$

We thus can detect $|G_{(u)}^\ell| \geq Y > \tau$ with probability $1 - 1/m^4$. □

We now present our key lemma.

Lemma 7 *For a contributing level ℓ, with probability $1 - 1/m^3$, we will reach Line 22 in Algorithm 3, or, we will have $|U| \geq \eta_\tau$.*

PROOF. It is easy to observe that in the worst case (w.r.t. the total communication cost), $\forall u \in \tilde{W}^\ell$ has $|G_{(u)}^\ell| = 1$. This is because (1) the sample probability for each item u is proportional to $|G_{(u)}^\ell|$; and (2) the communication spent on sampling sites and checking at Line $9 - 18$ for items in $\tilde{W}^\ell$ is negligible compared with the total budget t (defined at Line 3 of Algorithm 3):

$$\begin{aligned} & |U| \cdot k \cdot c_u \log N \\ \leq\ & \eta_\tau \cdot c_u k \log N \\ =\ & c_\eta \cdot 16 \log^2 m/\epsilon^2 \cdot \log(200(L+1)) \cdot c_u k \log N \\ =\ & o(t), \end{aligned}$$

Algorithm 3: Estimating $F_0(\tilde{W}^\ell)$ for an $\ell \in \{0, 1, \ldots, L\}$

1 $cost \leftarrow 0, U \leftarrow \emptyset$;
2 $\eta_\tau \leftarrow c_\eta \cdot 16 \log^2 m/\epsilon^2 \cdot \log(200(L+1))$ /* set $\delta = 1/(200(L+1))$ */;
3 $t \leftarrow c_t \cdot k/\epsilon^3 \cdot \log^3 k \log^2 m \log N$ /* c_t is a sufficiently large constant */;
4 **while** *($cost \leq t$) $\wedge$ ($|U| < \eta_\tau$)* **do**
5 the coordinator and sites generate a new sample (with replacement) $u \in W^\ell$;
6 $s \leftarrow 0$ /* number of sites contacted */;
7 $z \leftarrow 0$ /* number of items in $G_{(u)}^\ell$ found in sampled sites */;
8 **while** $s < k$ **do** /* test whether $u \in \tilde{W}^\ell$ */
9 the coordinator samples (without replacement) a random site $I \in [k]$, and sends u to site I ;
10 $s \leftarrow s + 1, cost \leftarrow cost + 1$;
11 **if** *$\exists v \in W_I^\ell$ such that $u \sim v$* **then**
12 $z \leftarrow z + 1$;
13 **if** $z > \tau$ **then**
14 mark u bad;
15 break /* do not satisfy item 3 in Definition 1 */;
16 **if** *$\exists v \in W_I^{\ell'}$ such that $\ell' > \ell$ and $u \sim v$* **then**
17 mark u bad;
18 break /* do not satisfy item 2 in Definition 1 */;
19 **if** *u is not marked bad* **then**
20 $U \leftarrow U \cup \{u\}$;
21 **if** $|U| \geq \eta_\tau$ **then**
22 run Algorithm 2 on U and output whatever Algorithm 2 outputs /* Since U can be kept at the coordinator locally, Algorithm 2 can be run on U without any communication */;
23 **else** output 0;

where $c_u \log N$ (for a small constant c_u) is the communication cost between Line $9 - 18$ (the coordinator sends u to the sampled site, and site gives feedback). We thus consider this worst case for simplicity.

Let $C_j^\ell = \{u \in S \mid |G_{(u)}^\ell| \in (2^{j-1}, 2^j]\}$ $(j = 0, 1, \ldots, \log k)$. Let $W_j^\ell = W^\ell \cap C_j^\ell$. In words, W_j^ℓ contains all items u in W^ℓ such that the corresponding group $G_{(u)}$ satisfies $|G_{(u)}^\ell| \in (2^{j-1}, 2^j]$. According to our worst case assumption, we have $\tilde{W}^\ell \subseteq W_0^\ell$.

Let $D^\ell \subseteq W^\ell$ be the set of sampled items at Line 5. Then $U = \tilde{W}^\ell \cap D^\ell$. Let $D_j^\ell = W_j^\ell \cap D^\ell$. We prove by contradiction. Suppose $|U| < \eta_\tau$, we will show that the total communication spent should be less than t with high probability. We need the following technical claim, which says that if the cardinality of U (useful samples from $\tilde{W}^\ell$) is small, then the cardinalities of all D_j are also small.

Claim 1 *If $|U| < \eta_\tau$, then $|D_j^\ell| < 2 \cdot 2^j \cdot L/\epsilon \cdot \eta_\tau$ with probability $1 - 1/m^4$ for each $j = 0, 1, \ldots, \log k$.*

PROOF. (for Claim 1) For each $j \in \{0, 1, \ldots, \log k\}$, we can assume that there are only two sets of items (overlap when $j = 0$), $\tilde{W}^\ell$ and W_j^ℓ, in W^ℓ, since a sample outside $\tilde{W}^\ell$ and W_j^ℓ will

Algorithm 4: Estimating F_0 for Distributed Noisy Datasets by Hierarchical Sampling

1 **for** $\ell = 0, 1, \ldots, L$ **do**
2 run Algorithm 3, and let z_ℓ be the output (i.e., an estimation of $F_0(\tilde{W}^\ell)$).
3 output $\sum_{\ell=0}^{L} z_\ell$.

not contribute to either U or D_j^ℓ. By the fact that the sampling probability for each item u is proportional to $|G_{(u)}^\ell|$, and $|\tilde{W}^\ell| \geq \epsilon/L \cdot n$ (definition of a contributing level ℓ), and the worst case assumption that $|G_{(u)}^\ell| = 1$ for each $u \in \tilde{W}^\ell$, we have that for each sample u,

$$\mathbf{Pr}\left[u \in \tilde{W}^\ell\right] = \frac{|\tilde{W}^\ell|}{|\tilde{W}^\ell \cup W_j^\ell|} \geq \frac{1 \cdot (\epsilon/L \cdot n)}{2^j \cdot n} = \frac{1}{2^j L/\epsilon}. \quad (3)$$

Let s be the total number of items we have sampled from $\tilde{W}^\ell \cup W_j^\ell$. Let $X_i = 1$ if the i-th sample in $|U \cup D_j^\ell|$ is in $\tilde{W}^\ell$, and $X_i = 0$ otherwise. Thus $\mathbf{E}[X_i] \geq 1/(2^j L/\epsilon)$ by Inequality (3) for each $i \in [s]$. Note that $U = \sum_{i\in[s]} X_i$, thus $\mathbf{E}[U] \geq s/(2^j L/\epsilon)$. By a Chernoff bound,

$$\mathbf{Pr}\left[|U| \geq \frac{\mathbf{E}[U]}{2}\right] \geq \mathbf{Pr}\left[|U| \geq \frac{s}{2 \cdot 2^j L/\epsilon}\right] \geq 1 - e^{-\frac{s}{8 \cdot 2 \cdot 2^j L/\epsilon}}.$$

Thus given $|U| < \eta_\tau$, we have $|U \cup D_j^\ell| = s < 2 \cdot 2^j L/\epsilon \cdot \eta_\tau$ with probability $(1 - e^{-\eta_\tau/8}) \geq (1 - 1/m^4)$. Therefore $|D_j^\ell| \leq |U \cup D_j^\ell| < 2 \cdot 2^j L/\epsilon \cdot \eta_\tau$ with probability at least $1 - 1/m^4$. □

By Lemma 6, Claim 1, and union bounds, with probability $1 - 1/m^4 \cdot t - 1/m^4 \cdot (\log k + 1) \geq 1 - 1/m^3$, the communication cost spent on Line $9 - 18$ (which is the asymptotically dominating cost) is bounded by

$$\begin{aligned}
&\sum_{j=0}^{\log k} \left(|D_j^\ell| \cdot 2\tau k/2^{j-1} \cdot c_u \log N\right) \quad \text{(Lemma 6)} \\
\leq\ &\sum_{j=0}^{\log k} \left((2 \cdot 2^j \cdot L/\epsilon \cdot \eta_\tau) \cdot 2\tau k/2^{j-1} \cdot c_u \log N\right) \quad \text{(Claim 1)} \\
\leq\ &8 c_u c_\eta \cdot k/\epsilon^3 \cdot \tau^2 \cdot L \log(200(L+1)) \cdot \log N \cdot (\log k + 1) \\
\leq\ &10000 c_u c_\eta \cdot k/\epsilon^3 \cdot \log^3 k \log^2 m \log N.
\end{aligned}$$

We get a contradiction by choosing a large enough constant c_t in the total communication budget t.

We run Algorithm 3 for each level $\ell = 0, 1, \ldots, L$, and the final output is the sum of outputs of the $L + 1$ runs. Our final algorithm is presented in Algorithm 4.

Theorem 3 *Algorithm 4 computes a $(1+\epsilon)$-approximation of $F_0(S)$ correctly with probability at least 0.99 in the distributed noisy data setting, using $\tilde{O}(k/\epsilon^3)$ bits of communication and $\tilde{O}(1)$ rounds.*

PROOF. For the correctness, by Lemma 7 and Theorem 2 (setting the error parameter $\delta = 1/(200(L+1))$), we know that the run of Algorithm 3 at a contributing level ℓ correctly computes a $(1+\epsilon)$-approximation of $F_0(\tilde{W}^\ell)$ with probability at least $1 - 1/m^3 - 1/(200(L+1)) \geq 1 - 1/(150(L+1))$. By a union bound over all (at most $L+1$) contributing levels, with probability at least $1 - 1/150$, we can compute a $(1+\epsilon)$-approximation of $\tilde{W}^\ell$ for all contributing ℓ. By Lemma 5, $|Q| = 0$ with probability at least $1 - 1/m$. Plugging inequality (2), with probability $1 - 1/150 - 1/m \geq 0.99$, we correctly compute a $(1 + O(\epsilon))$-approximation of $F_0(S)$.

For the communication cost, we run Algorithm 3 for $L + 1 = \tilde{O}(1)$ levels. The cost of each run is bounded by

$$\tilde{O}(k + k/\epsilon^3) + \tilde{O}(k/\epsilon^3) + \tilde{O}(\eta_\tau \cdot k) = \tilde{O}(k/\epsilon^3),$$

where the first term in LHS counts the cost of sampling items at Line 5; by Observation 1, the coordinator can sample s items with replacement using $\tilde{O}(k + s)$ bits of communication. The second term in LHS counts the cost of sampling the sites and performing the test if a sample $u \in \tilde{W}^\ell$ at Lines $8 - 18$, which is essentially bounded by t up to some $\tilde{O}(1)$ factor. The third term in LHS counts the cost of running Algorithm 2 where the cardinality of each group is upper bounded by τ. Summing up, the cost of $L + 1$ runs is bounded by $\tilde{O}(k/\epsilon^3)$.

For communication rounds, for each level ℓ, at Line 5, we can first sample $x_1 = t/k$ samples from W^ℓ, and find y_1 of them are in $\tilde{W}^\ell$. If $y_1 \geq \eta_\tau$ we stop, otherwise we sample another set of $x_2 = 2x_1$ samples from W^ℓ, and find y_2 of them are in $\tilde{W}^\ell$. If $y_1 + y_2 \geq \eta_\tau$ we stop, otherwise we keep doubling the sample size. For Line 9, we again use the doubling method to sample sites, that is, we start with sampling 1 site, and then keep testing and doubling the sample size if necessary. In this way the number of rounds spent on each sample level ℓ can be bounded by $\tilde{O}(1)$, thus also $\tilde{O}(1)$ for all $L + 1$ levels. Using the doubling method instead of sampling items and sites one by one will increase the total communication cost by at most a constant factor. □

Algorithm 5: L_0-Sampling for Distributed Noise-free Datasets

1 the coordinator and sites compute $\tilde{n}$, a $(1 + 0.1)$-approximation to n, using Algorithm 4;
2 the coordinator picks a random hash function $h : [N] \to [0, 2^{2+\log \tilde{n}} - 1]$ and sends to k sites;
3 each site i hashes all items in S_i using h, and sends $B_i = \{u \in S_i \mid h(u) = 0\}$ to the coordinator. Let $B = \bigcup_{i\in[k]} B_i$.
4 the coordinator outputs B if $|B| = 1$.

4. L_0-SAMPLING

An algorithm for F_0 can be used to design an algorithm for L_0-sampling. Let's first recall an algorithm, presented in Algorithm 5, for L_0-sampling in the noise-free setting (cf. [42]). This algorithm was originally designed for the data stream model, and we have modified/simplified it for our distributed setting when an approximation of the distinct elements n is known.

Lemma 8 (cf. [42]) *In Algorithm 5, $|B| = 1$ with probability at least $1/24$.*

In the noise-free setting, we run Algorithm 5 for C times in parallel for a sufficiently large constant C, pick the first instance that has a unique item u such that $h(u) = 0$, and output u as the outcome of the L_0-sampling. While in the noisy data setting, we again run Algorithm 5 for C times in parallel for a sufficiently large constant C, but with the following modifications:

1. Add Line 0: the coordinator (locally) randomly shuffles the order of the sites.

2. Replace Line 4 by Line $4'$: (*synchronization*, similar to the one in Algorithm 1) the coordinator checks for each $i \in [k]$, for each $u \in B_i$, if there exists a $j < i$ such that $v \in S_j$ and $v \sim u$. If yes, the coordinator deletes u from B_i. The check can be done by communicating u with site 1 to $i-1$ in order. Let $B'_1, \ldots, B'_k$ be the sets of $B_1, \ldots, B_k$ after the synchronization. The coordinator outputs $B' = \bigcup_{i\in[k]} B'_i$ if $|B'| = 1$.

The first random shuffling step is critical to bound the communication cost in the analysis.

Theorem 4 *There is an L_0-sampler that succeeds with probability at least 0.99 in the distributed noisy data setting, using $\tilde{O}(k)$ bits of communication and $\tilde{O}(1)$ rounds.*

PROOF. Our new algorithm is formed by adding Line 0 and replacing Line 4 with Line $4'$ in Algorithm 5, as described above. The correctness just inherits the one for the noise-free setting (Lemma 8), since in our algorithm we only consider a random but fixed item for each group G (the rest items in G will be deleted in the synchronization step at Line $4'$).

For the communication cost, Line 0 can be done locally at the coordinator. Line 1 needs $\tilde{O}(k)$ bits of communication by Theorem 3. Line 2 also needs $\tilde{O}(k)$ bits. For the cost at Line 3, we bound the size of $\sum_{i\in[k]} |B_i|$. Observe that for each $u \in S$, $\mathbf{Pr}[u \in B] = \mathbf{Pr}[h(u) = 0] = 1/2^{2+\log \tilde{n}} < 1/(2n)$, and we have $|S| = \sum_{i\in[k]} |S_i| \le kn$ (recall that each site can eliminate local duplicates at the beginning), thus $\mathbf{E}[\sum_{i\in[k]} |B_i|] \le 1/(2n) \cdot kn = k/2$. By a Chernoff bound, with probability at least $1 - 2^{-\Omega(k)}$, we have $\sum_{i\in[k]} |B_i| \le k$.

Now we bound the cost at Line $4'$. For each $u \in B$, let $I(u)$ be the smallest index (after the random shuffling in the newly added Line 0) such that there exists $v \in B_{I(u)}$ with $u \sim v$. This is well defined since $u \sim u$ (itself). The cost of Line $4'$ can be bounded $\sum_{u\in B} I(u)$.

For $j = 0, 1, \ldots, \log k$, let $C_j = \{u \in S \,|\, |G_{(u)}| \in (2^{j-1}, 2^j]\}$. Let $D_j = C_j \cap B$. We try to bound $\sum_{u\in D_j} I(u)$ for each j. First, since we have shuffled the sites randomly at the beginning, for a $u \in C_j$, $\mathbf{E}[I(u)] \le k/2^{j-1}$. On the other hand, $\mathbf{E}[D_j] \le 1/(2n) \cdot 2^j n = 2^{j-1}$, thus by Lemma 3, with probability at least $1 - 1/m^{10}$, $|D_j| \le 2 \cdot 2^{j-1} + C \log m$ for a sufficiently large constant C. Thus with probability $(1 - 1/m^{10} \cdot m) \ge (1 - 1/m^9)$, we have

$$\begin{aligned}\mathbf{E}\left[\sum_{u\in D_j} I(u)\right] &= \sum_{u\in D_j} \mathbf{E}[I(u)] \\ &\le (2 \cdot 2^{j-1} + C \log m) \cdot k/2^{j-1} \\ &\le 4Ck \log m.\end{aligned}$$

Summing over all classes $j = 0, 1, \ldots, \log k$, we have that with probability $1 - 1/m^8$,

$$\mathbf{E}\left[\sum_{j=0}^{\log k} \sum_{u\in D_j} I(u)\right] \le 8Ck \log m \log k.$$

By a Markov inequality and a union bound, we have $\sum_{u\in B} I(u) = \sum_{j=0}^{\log k} \sum_{u\in D_j} I(u) \le \tilde{O}(k)$ with probability at least 0.999.

Summing up all lines, the total communication cost is bounded by $\tilde{O}(k)$ with probability 0.99 (need to properly adjust the constant success probability of Algorithm 4 at Line 1).

Algorithm 6: Finding Heavy Hitters for Distributed Noisy Datasets

1 each site i computes a Misra-Gries sketch of size $\theta = c_\theta/\epsilon$ (for some sufficiently large constant c_θ), denoted by $\mathsf{MG}_i = \{(u_1, ct_1), \ldots, (u_\theta, ct_\theta)\}$, and sends MG_i to the coordinator;

2 the coordinator merges $\mathsf{MG}_1, \ldots, \mathsf{MG}_\theta$, by treating items belonging to the same group as one item, and adding up their ct's. At the end the coordinator outputs all items whose frequencies are more than $(\phi - \epsilon)m$ as (ϕ, ϵ)-heavy-hitter.

The number of rounds can be bounded by $\tilde{O}(1)$, by Theorem 3 and the fact that Line $4'$ can be done in $\tilde{O}(1)$ rounds using the doubling method when communicating u with site 1 to $i-1$ in order. □

5. HEAVY HITTERS, ENTROPY AND FREQUENCY MOMENTS

In this section we consider several statistical functions for which we can easily adopt existing algorithms designed for noise-free datasets, but one needs to find the right algorithms to extend.

5.1 Heavy Hitters

We note that the heavy-hitter problem is easy in the noisy data setting: Each site simply computes a Misra-Gries sketch [43] and sends it to the coordinator, and then the coordinator merges the k sketches and computes the set of heavy-hitters. See Algorithm 6 for details. The key feature here is that the Misra-Gries sketch simply consists of a list of (item ID, count) pairs, thus the coordinator can recognize if two items belong to the same group, and add up their counts.

Another algorithm that computes (ϕ, ϵ)-heavy-hitter is the simple sampling: The coordinator and sites sample $C \log N/\epsilon^2$ (for some large enough constant C) items from S, which is enough to estimate the cardinalities of all groups in $\mathcal{G}$ up to an additive error ϵm with success probability 0.99 (by a union bound).

Theorem 5 *There is an algorithm that correctly computes (ϕ, ϵ)-heavy-hitter with probability 0.99 in the distributed noisy data setting, using $\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$ bits of communication and 1 round.*

5.2 F_p ($p \ge 1$)

We observe that a very recent algorithm by Kannan et al. [34] for F_p in the coordinator model for noise-free datasets can be adapted for computing F_p in the noisy data setting. We comment that most F_p algorithms proposed in the streaming literature, e.g., [31], cannot be used here.

Denote $f(x) = x^p$ ($p \ge 1$). In the noise-free setting, let a_{iu} be the frequency of u in S_i, and in the noisy data setting, let $a_{iu} = |G_{(u)} \cap S_i|$. Let $[N]$ be the item universe. Let $C_i = \sum_{u\in[N]} f(a_{iu})$; $B_u = \sum_{i\in[k]} f(a_{iu})$; $A_u = f\left(\sum_{i\in[k]} a_{iu}\right)$. Let $A = \sum_{u\in[N]} A_u, B = \sum_{u\in[N]} B_u, C = \sum_{i\in[k]} C_i$.

For completeness, we first present the algorithm in [34] in Algorithm 7 (adapted to our notations), and then explain how to implement it in the noisy data setting.

Lemma 9 ([34]) *Algorithm 7 computes F_p ($p \ge 1$) correctly with probability at least 0.99 in the distributed noise-free setting, using $\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$ bits of communication and $O(1)$ rounds.*

Algorithm 7: [34] Estimating F_p $(p \geq 1)$ for Distributed Noise-free Datasets

1 the coordinator picks an i.i.d. sample Z_0 of $z_0 = k^{p-2}/\epsilon^3$ items $u \in S$, where each u is picked according to the probability $\frac{B_u}{B}$. More precisely, the coordinator first picks a site $i \in [k]$ according to the probability $\frac{C_i}{B}$, and then site i picks an item $u \in S_i$ according to the probability $\frac{f(a_{iu})}{C_i}$;
2 **foreach** $u \in Z_0$ **do**
3 the coordinator computes A_u and B_u by contacting k sites;
4 the coordinator computes $\rho = \frac{1}{z_0}\sum_{u\in Z_0}\frac{A_u}{B_u}$, and $\tilde{A} = \rho B$;
5 **if** $\tilde{A} \geq kB$ **then** output $\tilde{A}$ and terminate;
6 the coordinator picks an i.i.d. sample Z of $z = O(k^{p-1}(\log k)^2/\epsilon^3)$ items $u \in S$, each according to the probability B_u/B (same as Line 1);
7 let $\Gamma = \{k^{p-1}, e^{-\epsilon}k^{p-1}, e^{-2\epsilon}k^{p-1} \ldots, 1\}$;
8 **foreach** $\gamma \in \Gamma$ **do** (by the coordinator)
9 pick a subset $Y \subseteq Z$ of size $y = \Theta(\gamma(\log k)^2/\epsilon^3)$ uniformly at random;
10 **foreach** $u \in Y$ **do**
11 **for** $j = 1, \ldots, \kappa = \Theta(p \log k + \log(1/\epsilon))$ **do**
12 pick a set I of $q = \frac{k^{p-1}}{\gamma}$ of sites uniformly at random;
13 find all a_{iu} $(i \in I)$ and compute $x_j = \frac{k^p}{q^p}(\sum_{i\in I} a_{iu})^p$;
14 set $\tilde{A}_i$ to be the median of $x_1, \ldots, x_\kappa$;
15 **foreach** $u \in Y$ **do**
16 set $\tilde{B}_u = a_{i(u),u}$, where $i(u)$ denotes the index of the site where u is sampled in Line 6;
17 **foreach** $u \in Y$ *such that* $\tilde{A}_u/\tilde{B}_u \in [\gamma e^{-\epsilon}, 10k \log k \cdot \gamma)$ **do**
18 do an exact computation of A_u and B_u by contacting each site. Let $\phi_\gamma = \left|\{u \mid A_u/B_u \in [\gamma e^{-\epsilon}, \gamma)\}\right|$;
19 set $\varphi_\gamma = \phi_\gamma y/z$.
20 output $B\sum_{\gamma\in\Gamma}\varphi_\gamma\gamma$.

Algorithm 7 can be easily adapted to the noisy data setting. The general idea of the adaptation is that when we sample an item u or compute A_u, B_u for item u in Algorithm 7, we should sample or compute for the corresponding group $G_{(u)}$. We observe that in Algorithm 7:

1. When we want to compute A_u or B_u, we contact all sites.

2. When we need to sample an item with probability proportional to B_u/B, we first sample a site and then the site samples the item (Line 1). We can do this *without* knowing the frequency of that item.

These features enable us to run Algorithm 7 directly for groups instead of items. One can check that all steps in Algorithm 7 go through in the noisy data setting.

Theorem 6 *There is an algorithm that computes F_p $(p \geq 1)$ correctly with probability* 0.99 *in the distributed noisy data setting, using $\tilde{O}((k^{p-1}+k^3)/\epsilon^3)$ bits of communication and $O(1)$ rounds.*

Algorithm 8: Estimating the Empirical Entropy for Distributed Noisy Datasets

1 let $\gamma = c_\gamma \cdot 1/\epsilon^2 \log m$ for some large enough constant c_γ;
2 **for** $i = 1, \ldots, \gamma$ **do**
3 sample an item $u_i \in_R S$, and sample $r_{u_i} \in_R |G_{(u_i)}|$, by contacting the k sites;
4 compute $X_i = f(r_{u_i}) - f(r_{u_i} - 1)$;
5 sample an item $v_i \in_R S \backslash \{x \in S \mid x \sim u_i\}$, and sample $r_{v_i} \in_R |G_{(v_i)}|$, by contacting the k sites;
6 compute $Y_i = f(r_{v_i}) - f(r_{v_i} - 1)$;
7 use the Algorithm 6 (setting $\phi = 0.6, \epsilon = 0.01$) to test if there is a group $G \in \mathcal{G}$ s.t. $|G| \geq 0.6m$;
8 **if** *such a group G exists* **then**
9 compute $p_{\max} = |G|/m$ exactly;
10 **foreach** *sample* u_i $(i \in [\gamma])$ **do**
11 **if** $G = G_{(u_i)}$ **then** $Z_i \leftarrow Y_i$;
12 **else** $Z_i \leftarrow X_i$;
13 output $(1 - p_{\max})\frac{1}{\gamma}\sum_{i\in[\gamma]} Z_i + p_{\max}\log(1/p_{\max})$;
14 **else**
15 output $\frac{1}{\gamma}\sum_{i\in[\gamma]} X_i$;

5.3 Entropy

We can simply implement the AMS-sampling based algorithm in [13] for the streaming model in our distributed noisy data setting. We present our algorithm in Algorithm 8, which is basically a simplified version of the one in [13]. The main difference is that we always work on groups.

Theorem 7 *Algorithm 8 computes a $(1+\epsilon)$-approximation of the empirical entropy correctly with probability* 0.99 *in the distributed noisy data setting using $\tilde{O}(k/\epsilon^2)$ bits of communication and $O(1)$ rounds.*

PROOF. The correctness of Algorithm 8 directly follows from that in [13]. For the communication cost, the sampling part (Line 2-6) needs $\tilde{O}(k/\epsilon^2)$ bits of the communication, which is the dominating cost. □

Acknowledgments: The author would like to thank Funda Ergun, Dirk Van Gucht and Ke Yi for helpful discussions.

6. REFERENCES

[1] P. K. Agarwal, G. Cormode, Z. Huang, J. M. Phillips, Z. Wei, and K. Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26, 2013.

[2] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012.

[3] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14, 2012.

[4] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[5] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[6] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, pages 586–597, 2002.

[7] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117, 2008.

[8] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *ICALP (1)*, pages 95–106, 2009.

[9] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*, pages 1–10, 2002.

[10] R. Bardenet and O.-A. Maillard. Concentration inequalities for sampling without replacement. *arXiv preprint arXiv:1309.4029*, 2013.

[11] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1):255–276, 2009.

[12] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *ICDM*, pages 47–58, 2006.

[13] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Transactions on Algorithms*, 6(3), 2010.

[14] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[15] J. Chen and Q. Zhang. AMS-Sampling in Distributed Monitoring, with Application to Tracking Entropy. *Manuscript*, 2014.

[16] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[17] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms*, 7(2):21, 2011.

[18] X. L. Dong and F. Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[19] A. Drucker, F. Kuhn, and R. Oshman. On the power of the congested clique model. In *PODC*, pages 367–376. ACM, 2014.

[20] W. W. Eckerson. Data quality and the bottom line. *TDWI Report, The Data Warehouse Institute*, 2002.

[21] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[22] L. English. Plain English on data quality: Information quality management: The next frontier. *DM Review Magazine*, Apr. 2000.

[23] P. Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[24] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *SPAA*, pages 281–291, 2001.

[25] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the mapreduce framework. In *ISAAC*, pages 374–383, 2011.

[26] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, pages 127–138, 1995.

[27] T. N. Herzog, F. J. Scheuren, and W. E. Winkler. *Data quality and record linkage techniques*, volume 1. Springer, 2007.

[28] Z. Huang, K. Yi, Y. Liu, and G. Chen. Optimal sampling algorithms for frequency estimation in distributed data. In *INFOCOM*, pages 1997–2005, 2011.

[29] Z. Huang, K. Yi, and Q. Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *PODS*, pages 295–306, 2012.

[30] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[31] P. Indyk and D. P. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208, 2005.

[32] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SDM*, 2005.

[33] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52, 2010.

[34] R. Kannan, S. Vempala, and D. P. Woodruff. Principal component analysis and higher correlations for distributed data. In *COLT*, pages 1040–1057, 2014.

[35] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *SODA*, pages 938–948, 2010.

[36] H. Klauck, D. Nanongkai, G. Pandurangan, and P. Robinson. The distributed complexity of large-scale graph processing. 2015.

[37] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, pages 802–803. ACM, 2006.

[38] N. Koudas and D. Srivastava. Approximate joins: Concepts and techniques. In *VLDB*, pages 1363–1363. VLDB Endowment, 2005.

[39] P. Koutris and D. Suciu. Parallel evaluation of conjunctive queries. In *PODS*, pages 223–234, 2011.

[40] C. Lenzen. Optimal deterministic routing and sorting on the congested clique. In *PODC*, pages 42–50. ACM, 2013.

[41] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. Mst construction in o (log log n) communication rounds. In *SPAA*, pages 94–100. ACM, 2003.

[42] A. McGregor. Lecture notes. Available at `http://people.cs.umass.edu/~mcgregor/courses/CS711S12/index.html`, 2012.

[43] J. Misra and D. Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[44] A. E. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *DMKD*, pages 23–29, 1997.

[45] B. Patt-Shamir and M. Teplitsky. The round complexity of distributed sorting. In *PODC*, pages 249–256. ACM, 2011.

[46] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *SODA*, pages 486–501, 2012.

[47] P. Singla and P. Domingos. Entity resolution with markov logic. In *ICDM*, pages 572–582, 2006.

[48] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.

[49] D. P. Woodruff and Q. Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960, 2012.

[50] D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. In *DISC*, pages 16–30, 2013.

[51] D. P. Woodruff and Q. Zhang. An optimal lower bound for distinct elements in the message passing model. In *SODA*, pages 718–733, 2014.

APPENDIX

A. A FEW REMARKS ON THE MODEL AND ASSUMPTIONS

Remark 2 (The Local Comparison Metric) For a specific application, the comparison metric can simply be a distance function $d(\cdot,\cdot)$. For example, if items are points in the plane then we can use the Euclidean distance to measure their similarities. However, we choose not to specify the metric here, for two reasons: First, using a general metric allows our algorithms to work with *any* distance/similarity functions, such as Jaccard distance, cosine distance, edit distance, etc. – in other words, our algorithms are *generic*; we have decoupled the algorithm design for statistical problems (our focus here) and similarity measurements. Second, for certain applications, it is very hard to assume that similarities between items can be expressed by a well-known distance function. For example, one may think edit distance is a good measurement for comparing the similarities between corporation names, but it turns out that "AT&T Corporation" is closer to "IBM Corporation" than "AT&T Corp" [38] under edit distance! The situations could be more complicated for big objects such as images/musics/videos. Therefore it is better to allow user-defined (domain knowledge based) comparison metrics at the runtime of our algorithms.

In this paper we use pairwise comparison because we feel that other approaches, such as relational approaches [6, 32] and collective approaches [12, 47] mentioned before, though sometimes very effective in (small scale) data de-duplication, may not be used for the design of sublinear communication algorithms in the large-scale distributed setting.

Remark 3 (Transitivity) As for the transitivity assumption, one may come up with the following problematic example: we have $a \sim b, b \sim c, \ldots, y \sim z$, however, $a \not\sim z$. We can of course select distance functions carefully to avoid such situations, for example, trying to find a distance function $d(\cdot,\cdot)$ such that $d(u,v) < \beta$ if $u \sim v$ and $d(u,v) \geq \alpha$ ($\alpha > 2\beta > 0$) if $u \not\sim v$. But we do agree that it is sometimes hard to find a good distance function that can completely avoid such "chain phenomena". However, if we can remove a set of items O (call them "outliers") so that the resulting items can be well-partitioned to groups, and $|O| + |\mathcal{N}(O)|$ is small where $\mathcal{N}(O) = \{v \mid \exists u \in O \text{ s.t. } v \sim u\}$ are the "touching points" of outliers in well-shaped groups, then our sampling based algorithms should still perform well. The reason is that with a good probability, none or few of the outliers O and their neighbors $\mathcal{N}(O)$ will be sampled, thus we can simply ignore them. Note that the other items will never be compared with outliers in O. Take F_0 and Algorithm 2 (Section 3.2.2) for example, if $|O|+|\mathcal{N}(O)| \leq \epsilon F_0(S\backslash O) \approx \epsilon \cdot \left(\frac{1}{\eta_k}\sum_{i\in[\eta_k]} X_i\right) m$, then the estimator in Algorithm 2 (Line 5) still gives a $(1+\epsilon)$-approximation with probability 0.99 (by changing the constants slightly). On the other hand, if we cannot find a small set of outliers, then maybe the dataset itself is difficult for entity resolution.

There do exist approaches that do not assume transitivity. One way is to assume so-called ICAR properties [11]. But this pairwise comparison based approach is iterative in nature and is designed for listing all distinct entities instead of our much less expensive statistical estimations. Another way is to use clustering based approaches. For example, the paper by Alion et al. [4] formulated a clustering problem trying to minimize the global inconsistency of the final clustering using linear programming. But the clustering problem is NP-hard and their algorithms are centralized. To sum up, it is not clear how to design communication and round efficient algorithms in the distributed setting using these alternative approaches, and we leave it as a future work.

Remark 4 (Magic Hash Function) Another question is whether there exists some hash functions that can simply map items in a same group into a same bucket. If this is true then similar items can just be treated as one item and our problems will become trivial, that is, all previous algorithms for the noise-free data setting directly apply. We argue that first, one cannot hope to get a magic hash function that works for all distance functions. Second, it could be very difficult to design such a hash function that works for some practical but complicated distance functions, such as cosine distance plus tf-idf vector space model for comparing document similarity. Third, one may think of using locality sensitive hashing (LSH) [7, 30], for which we explain a bit more.

An (ℓ, u, p_ℓ, p_u)-sensitive LSH family $\mathcal{H}$ can only guarantee that $\mathbf{Pr}_{h\in_R\mathcal{H}}[h(a) = h(b)] > p_\ell$ if $d(a,b) < \ell$; and $\mathbf{Pr}_{h\in_R\mathcal{H}}[h(a) = h(b)] < p_u$ if $d(a,b) > u$ ($a \in_R A$ means a is picked uniformly at random from set A). Therefore items in a same group can still land in different buckets. For example, if p_ℓ, p_u are both constants, then a *constant fraction* of pairs of similar items will go to different buckets (think about imposing a random grid on points in the Euclidean plane), thus one cannot hope for a $(1+\epsilon)$-approximation. One may want to use the standard AND-OR trick to amplify the gap between p_ℓ and p_u, but we notice that the AND-OR trick cannot be applied in the distributed setting without using a significant amount of communication.

To sum up, a magic hash function, if it exists, must be metric-dependent and will need a large number of bits to describe (in the extreme case, the range of the hash function is just the set of all representative distinct universe elements, and one can "hash" each item to their closest representative universe element), and thus cannot be communication-efficiently applied to the coordinator model.

Remark 5 (Local Entity-Resolution with A Shared Vocabulary) The final remark is about whether the parties can resolve locally its own entity-resolution problem by considering a shared vocabulary/ontology. A shared vocabulary is essentially equivalent to a magic hash function discussed above, which is unlikely to exist or is communication-expensive to compute centrally and then send to each party. One may think such a shared vocabulary can be precomputed/stored in each party. This may be possible if data items are short strings (e.g., names and short addresses). But it should be infeasible for documents/images/videos since the size of the "vocabulary" would be prohibitive.