Finding Items Associated with Varied Members in A Pairwise Data Stream

Heejae Yim* Norikazu Takahashi†‡ Yukiko Yamauchi* Shuji Kijima*‡
Masafumi Yamashita*‡

June 14, 2013

Abstract

For large data, stream algorithm is an effective method to get useful information. Several stream algorithms have been developed such as Lossy Counting and Distinct Sampling for stream data of items. This paper is concerned with streams composed of *multi-dimensional elements*. We present an algorithm which handles a stream of pair elements (x, y), to search x that appears with a lot of distinct y. The algorithm is based on two stream algorithms, Lossy Counting and Distinct Sampling. We also show some theoretical analysis of the algorithm concerning approximate guaranty, and memory space for computing.

Keywords: Stream Algorithm, Lossy Counting, Distinct Sampling, Distinct Value, Variety.

1 Introduction

In recent years, the data generated from the Internet is growing fast with the development of information technology. It is not easy to deal with very large data generated rapidly. This paper discusses stream algorithm that is a method to obtain useful information from large data. Stream algorithm is data processing whose input data is a sequence of items over stream under the limited memory space. In other word, stream algorithm can pick out the required items, even in insufficient memory space.

There are several stream algorithms such as Lossy Counting[1], Distinct Sampling[2] etc. Lossy Counting is an algorithm to find frequent items computing ϵ-deficient synopses over data stream. Distinct Sampling is an algorithm to estimate the number of distinct values.

Both algorithms deal with a stream consisting of single elements. However, there are many streams consisting of pairs of items from a set X and Y respectively. For example, suppose a stream consisting of (sender, receiver) on e-mail, a sender who sends e-mails to varied receiver may be a spam sender. For another example, the Internet traffic data is a stream making up source address and destination address. To the best of our knowledge, no research has yet been carried out to handle the algorithm for these streams.

Out Results

In this paper, we present an algorithm, which handles a stream of pair of elements (x, y), to search x that appear with a lot of distinct y. Our algorithm is based on two stream algorithm, Lossy Counting[1] and

*Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
{heejae.yim,yamauchi,kijima,mak}@inf.kyushu-u.ac.jp
†Graduate School of Natural Science and Technology, Okayama University, Okayama, Japan
takahashi@cs.okayama-u.ac.jp
‡Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka, Japan

Distinct Sampling[2]. As a result, it is found that new algorithm can detect x that appears with many distinct y over data stream of (x, y), using a small memory space.

Section 2 briefly reviews Lossy Counting and Distinct Sampling. Section 3 we explain our problem. Section 4 we present a new algorithm for a stream of the pair data. Finally the performance of the proposed algorithm is analyzed theoretically in Section 5.

2 Preparation

2.1 Lossy Counting

We explain about Lossy Counting algorithm, which Manku and Motwani[1] has proposed. Input data is a stream S of a single element x_i: $S = (x_i)_{i=1}^N (x_i \in X)$. This algorithm computes the frequency of each element as i-th data arrives, and finally wants to find frequent elements. Denote support value γ and error ϵ. Each element x has entry (x, f, Δ), where f is estimated frequency and Δ is the maximum possible error in f. Let be $\mathcal{D}$ is a set of entries (x, f, Δ). Output is elements x such that $f \geq (\gamma - \epsilon)N$. The entering stream is divided into bucket's width $w = \lceil \epsilon^{-1} \rceil$. Bucket is labeled and the current bucket id is $b_{current}$. The data of the i-th will enter the bucket $\lceil \frac{i}{w} \rceil$. Lossy Counting algorithm is given in Algorithm 1 and satisfies as followings.

Algorithm 1 Lossy Counting

```
Input: data stream $S = (x_i)_{i=1}^N \in X^N$
 1: initialize $\mathcal{D} \leftarrow \emptyset$, and $i \leftarrow 1$
 2: for $i = 1$ to $N$ do
 3:   if $x_i$ does not exist in $\mathcal{D}$ then
 4:     add $(x_i, 1, b_{current})$ to $\mathcal{D}$
 5:   else
 6:     increment $f$ of $(x_i, f, \Delta)$ by one
 7:   end if
 8:   if $i \equiv 0 \pmod{w}$ then
 9:     delete $(x, f, \Delta)$ from $\mathcal{D}$ if $f \leq b_{current} - \Delta$ holds for the item $x$
10:   end if
11: end for
12: return $(x, f, \Delta)$ satisfies that $f \geq (\gamma - \epsilon)N$
```

Theorem 2.1. *[1] Algorithm 1 output x whose true frequency $f_e \geq \gamma N$, and does not output one whose f_e is less than $(\gamma - \epsilon)N$. If $(x, f, \Delta) \in \mathcal{D}$, then $f_e - \epsilon N \leq f \leq f_e$.*

Theorem 2.2. *[1] The number of entires (x, f, Δ) in Lossy Counting is at most $\epsilon^{-1} \log(\epsilon N)$.*

2.2 Distinct Sampling

We briefly introduce Distinct Algorithm[2]. The purpose of the algorithm is to find the number of distinct elements over a single data stream. Let be input stream $S = (x_i)_{i=1}^N$, consist of $x_i \in X = \{1, 2, ..., A\}$. It is assumed that the memory space is limited, so it is impossible to record all elements. Therefore we record only some elements using *die-hash function* $dh_{\alpha,\beta}$.

Definition 2.3. [2] (*die-hash function* $dh_{\alpha,\beta}$) Let A be a prime number, and $\alpha, \beta \in [1, A]$. Let m be $2^{m-1} \leq A < 2^m$. For any $x \in \{1, 2, .., A\}$, *die-hash function* $dh_{\alpha,\beta}(x)$ is defined by

$$dh_{\alpha,\beta}(x) = m - \lfloor \lg f_{\alpha,\beta}(x) \rfloor \tag{1}$$

where $f_{\alpha,\beta}(x) = \alpha x + \beta \bmod A$.

If we choose α, β uniformly at random from $[1, A]$, the following probability is satisfied for any $x \in \{1, 2, .., A\}$.

$$\Pr[dh(x) \geq l] = \begin{cases} 1 & (l = 1) \\ 2^{m+1-l}/A & (l \geq 2) \end{cases} \tag{2}$$

Let Y be memory space for sampling elements and its size is M. l is select level about die-hash function. Distinct Sampling algorithm is given in Algorithm 2. This algorithm returns the estimated variety by multiplying $|Y|$ and the reciprocal of sampling probability. Accuracy guarantees for Distinct Sampling, difference between the estimated value and true one, is calculated by using Chebyshev's inequality.

Theorem 2.4. *[2] With respect to Algorithm 2, let the true number of distinct elements be n, the estimated one be v. For any $t > 0$,*

$$\Pr[|v - n| \geq tn] < \frac{2}{t^2 M}. \tag{3}$$

Algorithm 2 Distinct Sampling

```
Input: data stream $S = (x_i)_{i=1}^N \in \{1, 2, .., A\}^N$
 1: choose $\alpha, \beta$ randomly from $[1, A]$
 2: initialize $Y \leftarrow \emptyset$, $i \leftarrow 1$, and $l \leftarrow 1$
 3: for $i = 1$ to $N$ do
 4:     if $dh_{\alpha,\beta}(x_i) \geq l$ then
 5:         add $x_i$ to $Y$
 6:     end if
 7:     if $|Y| = M$ then
 8:         delete $x$ from $Y$ if it satisfies that $dh_{\alpha,\beta}(x) = l$
 9:         increment $l$ by one.
10:     end if
11: end for
12: return the estimated variety $\begin{cases} |Y| & (l = 1) \\ |Y|M2^{l-m-1} & (l \geq 2) \end{cases}$
```

3 Problem Definition

In this section, we define our problem. Let $X = \{1, 2, ..., A\}$ and $Y = \{1, 2, ..., B\}$ be a finite set of items, respectively. An input stream S consists of a pair of items respectively in X and Y, i.e., $S = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$. We define $v_i(x)(1 \leq i \leq N)$ is the number of distinct y that have appeared with x on the data from 1 to i, i.e.,

$$v_i(x) = |Y_i(x)|, \ where \ Y_i(x) = \{y \in Y \mid \exists j \in \{1, 2, \ldots, i\}, (x_j, y_j) = (x, y)\}. \tag{4}$$

For example, let x is a sender and y is a receiver of an e-mail, then stream data S is the continuously arranged data composed of (sender, receiver). $v_i(x)$ means that how many people x has sent e-mail to, on the data from 1 to i. Notice that even thought x has sent e-mail to one receiver many times, $v(x)$ is counted as one. If x has sent e-mail to only one receiver i times on the data from 1 to i, $v_i(x)$ is 1. If x has sent e-mail to i people once, $v_i(x)$ is i.

In this paper, we suppose that we cannot reserve all data from stream S, because the length of stream S is very large. Under this condition, we want to find $x \in X$ such that $v_N(x)$ is greater than or equal to some value. To be more precise, the problem is to find $x \in X$ such that

$$v_N(x) \geq \gamma N \tag{5}$$

holds, where γ is positive constant. For example, suppose S consists of (sender, receiver) on e-mail. To find x such that $v_N(x) \geq \gamma N$ means that we want to find someone who send e-mail to greater than or equal to γN people. However, in this problem, we suppose that information about senders and receivers is very large, thus we cannot record all in memory space.

4 Algorithm

Algorithm 3 Algorithm for finding elements x associated with various items

```
Input: data stream $S = \{(x_i, y_i)\}_{i=1}^N$
 1: set parameters $\gamma$ ,$\epsilon \in (0,1)$
 2: select $\alpha, \beta$ for the hash function $dh_{\alpha,\beta}(y)$
 3: initialize $\mathcal{D} \leftarrow \emptyset$, $i \leftarrow 1$, $b_{current} \leftarrow 1$
 4: for $i = 1$ to $N$ do
 5:   if $x_i$ is does not exist in $\mathcal{D}$ then
 6:     insert $(x_i, 1, b_{current} - 1, 1, \{y_i\})$ in $\mathcal{D}$
 7:   else
 8:     if $dh_{\alpha,\beta}(y_i) \geq l(x_i)$ and $y_i \notin \hat{Y}(x_i)$ then
 9:       add $y_i$ to $\hat{Y}(x_i)$ of $(x_i, \hat{v}(x_i), \Delta(x_i), l(x_i), \hat{Y}(x_i)) \in \mathcal{D}$
10:     end if
11:     if $|\hat{Y}(x_i)| = M$ then
12:       delete $y$ such as $dh_{\alpha,\beta}(y) = l(x_i)$ from $\hat{Y}(x_i)$
13:       increment $l(x_i)$
14:     end if
15:   end if

16:   update $\hat{v}_i(x)$ by the following equation
        $$\hat{v}(x_i) = \begin{cases} |\hat{Y}(x_i)|, & \text{when } l(x_i) = 1 \\ |\hat{Y}(x_i)| B 2^{l(x_i)-m-1}, & \text{else } l(x_i) \geq 2 \end{cases} \tag{6}$$
17:   if $i \equiv 0 \pmod{\lceil \frac{1}{\epsilon} \rceil}$ then
18:     eliminate $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ from $\mathcal{D}$ if $x$ satisfies
          $$\hat{v}(x) \leq b_{\text{current}} - \Delta(x) \tag{7}$$
19:     increment $b_{current}$ by one
20:   end if
21: end for
22: return $x$ satisfies that $\hat{v}(x) \geq (\gamma - \epsilon)N$
```

We now present a new algorithm to solve the problem. Let $M (\leq B)$ be a positive number. M is the memory space for memorizing y. User specifies a support value γ and error ϵ.

We make the algorithm by combining Lossy Counting[1] and Distinct Sampling[2]. Lossy Counting deletes infrequent elements in each bucket. Distinct Sampling uses die-hash function to sample some elements and estimates the number of distinct elements. Our algorithm is based on bucket and die-hash function. First, estimate $v(x)$, the number of distinct y which appears with x, using die-hash function. Next, delete x such that $v(x)$ is less than some value in each bucket.

We will describe the data structure, called entries. The algorithm renews entries, a set of five pairs, whenever a pair (x_i, y_i) arrives. Let $\mathcal{D}$ be a set of entries $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$. $\hat{v}(x)$ is the estimated value of $v(x)$. $\Delta(x)$ is for deletion after bucket end. $l(x)$ is select level for using die-hash function. $\hat{Y}(x)$ is a set for estimate $Y(x)$ ($Y(x)$ is a set of y). It assumes that the maximum number of $\hat{Y}(x)$ is M. $x, \hat{v}(x), \Delta(x)$ are obtained from Lossy Counting and $l(x), \hat{Y}(x)$ are obtained from Distinct Sampling. Notice that Lossy Counting uses frequency f, but this algorithm records $\hat{v}(x)$, which is the estimated number of distinct y which appear with x. The proposed algorithm is given in Algorithm 3

We use bucket in the same way as Lossy Counting. One bucket has ϵ^{-1} data. The current bucket is $b_{current}$, and the last bucket is $\lceil \frac{N}{w} \rceil$. In Step 2, we select a die-hash function by choosing $\alpha, \beta \in \{1, 2, ..., B\}$ uniformly at random. This die-hash function is the same one in Distinct Sampling. Let $dh_{\alpha,\beta}(y)$ denote the die-hash function. Let m be $2^{m-1} \leq B < 2^m$. Then,

$$dh(y) = dh_{\alpha,\beta}(y) = m - \lfloor \lg f(y) \rfloor \tag{8}$$

where $f(y) = f_{\alpha,\beta}(y) = \alpha y + \beta \bmod B$.

From Step 4, we start to read data. First, if x_i does not exist in $\mathcal{D}$, insert a new entry about x_i in $\mathcal{D}$. If x_i already exists, we want to add y to $Y(x_i)$ and calculate $v_i(x)$. However, since the memory space is limited($|\hat{Y}| \leq M$), we pick up only y whose hash value is greater than or equal to select level $l(x)$. Then, add them to $\hat{Y}(x)$. We can estimate the number of distinct y by Eqn. (6).

We delete data in two ways. One is from Step 11 to Step 14. If $\hat{Y}$ is full, delete y whose hash value is $l(x_i)$. Then, the vacant space is created in $\hat{Y}$, because hash value of almost half of y in $\hat{Y}$ is $l(x)$. Another delete is from Step 17 to Step 20. After the last data arrive to each bucket, we remove entires such that satisfy Eqn. (7). Finally, the algorithm outputs x such that $\hat{v}(x) \geq (\gamma - \epsilon)N$.

5 Analysis of Our Algorithm

The purpose of the proposed algorithm is to find x such that $v_N(x) \geq \gamma N$. However, we cannot guarantee output, because we estimate $v_N(x)$ by $\hat{v}(x)$ probabilistically. We call error that some x which satisfy $v_N(x) \geq \gamma N$, but not included in output. There are two reasons why error occurred:

1. We record y, using die-hash function generated by α, β at random.

2. We remove entires whose $\hat{v}(x)$ is small by Step 18. However, if we accidentally remove entires originally not deleted by the difference between the estimated value and true one, error may become great.

Concerning the above issue 1, Distinct Sampling algorithm calculates the difference between true and estimated value using Chebyshev's inequality. However, we also use Lossy Counting, so error can occur by the issue 2. As a result, error would occur according to hash function selected randomly, γ, ϵ, and memory size M. We will define some set of x to discuss about these and show some theorem.

$\Psi = \{x \mid x \text{ appear in stream}\}$
$W = \{x \mid v_N(x) \geq \gamma N\}$
$\bar{W} = \{x \mid v_N(x) < \gamma N\}$
$U = \{x \mid \forall k \in \{1, 2, ..., \epsilon N - 1\}, v_{\epsilon^{-1}k}(x) > k - \Delta_{\epsilon^{-1}k}(x)\}$

$\bar{U} = \{x \mid \exists k \in \{1, 2, ..., \epsilon N - 1\}, v_{\epsilon^{-1}k}(x) \leq k - \Delta_{\epsilon^{-1}k}(x)\}$

Note that $\Psi = W \cup \bar{W}$ holds. U is a set of x whose $v(x)$ not satisfy the delete condition Eqn. (7) every bucket. The number of data in a bucket is ϵ^{-1}, so that $v_{\epsilon^{-1}k}(x)$ is $v(x)$ when the last (x, y) arrive to bucket k.

Theorem 5.1. *If M is large enough to record all y, Algorithm 3 outputs $x \in W$. Then, Algorithm 3 uses entires $(x, \hat{v}, \Delta, l, \hat{Y})$ at most $\epsilon^{-1} \lg(\epsilon N)$.*

Theorem 5.1 implies that Algorithm 3 has very similar result with Lossy Counting, when M is large enough to record all $y(v(x) \leq M)$. We prove this by using that $v(x)$ and $\hat{v}(x)$ are equal when $l(x) = 1$ by Eqn. (6). However, it is often that M is insufficient to record all y for large data. With limited memory space M, to analyze x which has deleted on bucket is difficult, because we don't remember past information about the deleted x. Therefore, we consider $x \in W \cap U$ first as follows.

Theorem 5.2. *Let $\Pr[E_o]$ denote the probability that Algorithm 3 outputs $x \in W \cap U$, then*

$$\Pr[E_o] \quad \geq \quad \prod_{j=k}^{\epsilon N-1} \Big\{1 - \frac{2}{M}\Big(\frac{v_{\epsilon^{-1}j}(x)}{v_{\epsilon^{-1}j}(x) - j + k - 1}\Big)^2\Big\}\Big\{1 - \frac{2}{M}\Big(\frac{v_N(x)}{v_N(x) - (\gamma - \epsilon)N}\Big)^2\Big\}, \tag{9}$$

k is the first bucket id, where x appears.

More generally, we have the following corollary.

Corollary 5.3. Set $M \geq \frac{8\epsilon N}{c\delta^2}$, for $c \in (0, 1]$ and $\delta \in (0, 1]$, the probability $\Pr[E'_o]$ that Algorithm 3 outputs $x \in W' \cap U'$ satisfies,

$$\Pr[E'_o] \geq (1 - \frac{1}{e})^c,$$

where $W' = \{x | v_{\gamma N}(x) \geq \gamma N(1+\delta)\}$, $U' = \{x | \forall j \in \{1, ..., \epsilon N - 1\}, v_{\epsilon^{-1}k}(x) > (k - \Delta_{\epsilon^{-1}k}(x))(1+\delta)\}$.

A sketch of proof of Theorem 5.2. Define the following events.

E_o : For $x \in W \cap U$, $\hat{v}(x) \geq (\gamma - \epsilon)N$
E_d : For $x \in W \cap U$, $\exists k \in \{1, ..., \epsilon N - 1\}$, $\hat{v}_{\epsilon^{-1}k}(x) \leq k - \Delta_{\epsilon^{-1}k}(x)$
$\bar{E}_d$: For $x \in W \cap U$, $\forall k \in \{1, ..., \epsilon N - 1\}$, $\hat{v}_{\epsilon^{-1}k}(x) > k - \Delta_{\epsilon^{-1}k}(x)$

E_d is that x has been removed in some bucket at least one time, and $\bar{E}_d$ is that x hasn't been removed in any bucket. Notice that $x \in W \cap U$ have $v_i(x)$ which not satisfies the delete condition Eqn. (7). Therefore E_d implies that x is accidentally removed. We can bound $\Pr[E_o]$ by

$$\Pr[E_o] = \Pr[E_o, E_d] + \Pr[E_o, \bar{E}_d] \geq \Pr[E_o, \bar{E}_d] = \Pr[\bar{E}_d] \Pr[E_o | \bar{E}_d].$$

$\Pr[\bar{E}_d]$ and $\Pr[E_o | \bar{E}_d]$ can be calculated by Chebyshev's inequality. □

A sketch of proof of Corollary 5.3. We can bound the probability generally if we ensure M. Assume that $x \in W' \cap U'$ satisfy $v(x) \geq (1 + \delta)t$ for some t. Then,

$$\frac{2}{M}(\frac{v(x)}{v(x) - t})^2 \leq \frac{2}{M}(\frac{(1+\delta)t}{\delta t})^2 = \frac{2}{M}(\frac{1+\delta}{\delta})^2 \leq \frac{8}{M\delta^2}.$$

For $M \geq \frac{8\epsilon N}{c\delta}$, $1 - \frac{2}{M}(\frac{v(x)}{v(x)-t})^2 \geq 1 - \frac{c}{\epsilon N}$. Then, we can bound Eqn. (9) as follows.

$$\Pr[E'_o] \geq \prod_{j=1}^{\epsilon N-1} (1 - \frac{c}{\epsilon N}) \cdot (1 - \frac{c}{\epsilon N}) = (1 - \frac{c}{\epsilon N})^{\epsilon N} = \{(1 - \frac{c}{\epsilon N})^{\frac{\epsilon N}{c}}\}^c \geq (1 - \frac{1}{e})^c$$

□

For example, let $c = 0.01$, then $\Pr[E'_o] \geq (1 - \frac{1}{e})^{0.01} = 0.9954$.

6 Conclusions

We proposed the algorithm to find elements that appear with a lot of elements over data stream. We presented theoretical analysis on this algorithm. In this paper, we did not show experimental results using an artificial data and the Internet traffic data. However, the algorithm can also be applied at the large data stream composed of pair elements.

The proposed algorithm is based on two stream algorithms. An improvement of the algorithm by using other hash function, or changing method to delete in bucket is a future work. An contention of the problem to the stream of multi-dimensional elements is another work.

Acknowledgment

This work is partly supported by Proactive Response Against Cyber-attacks Through International Collaborative Exchange (PRACTICE), Ministry of Internal Affairs and Communications, Japan.

References

[1] G.S.Manku and R.Motwani, Approximate frequent counts over data streams, Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), pp.346–357.

[2] P.B.Gibbons, Distinct sampling for highly-accurate answers to distinct values queries and event reports, Proceedings of the 27th International Conference Very Large Data Bases (VLDB 2001), pp.541–550.