

# Finding Frequent Items in Sliding Windows with Multinomially-Distributed Item Frequencies\*

Lukasz Golab, David DeHaan, Alejandro López-Ortiz

Erik D. Demaine

*University of Waterloo, Canada*  
{lgolab,dedehaan,alopez-o}@uwaterloo.ca

*M.I.T., USA*  
edemaine@mit.edu

## 1 Introduction

In this paper, we present an algorithm for identifying frequently occurring items within a sliding window of the last  $N$  items seen over an infinite data stream, given the following constraints.

- The relative frequencies of the item types can vary over the lifetime of the stream, provided that they vary sufficiently slowly that for any sliding window of  $N$  tuples, with high probability the window could have been generated by a multinomial distribution. We refer to this as the *drifting distribution model* in the full version of this paper [1].
- The entire sliding window does not fit in the available memory (otherwise, we could simply count all the distinct item types and return those whose frequencies exceed some threshold).
- The stream may arrive at a high rate, so only a constant number of operations (amortized) is allowed for the processing of each item.

The basic idea of our algorithm is to divide the sliding window into  $n$  equally-sized, contiguous partitions (called *basic windows* in [3]), with each partition storing only the identities of those item types which occur with a relative frequency of at least  $1/m$  within this particular basic window. Rather than advancing the window whenever a new tuple arrives, we wait until the most recent partition is full, and advance the window periodically by replacing the oldest partition with the newest; this occurs every  $b = N/n$  tuples. However, in the general case, there is no obvious way of computing which item types occur with a relative frequency that exceeds the threshold given only the identities of frequently occurring items within individual basic windows. For example, if each basic window

only stores the identities of its top three categories, we would completely ignore a frequent item type that consistently ranks fourth in each basic window. In what follows, we will show how to bound the error of the above technique under the assumption that the item types are multinomially distributed. More details may be found in the full version of this paper [1].

## 2 The Algorithm

We begin with a precise definition of our algorithm. Recall that results are refreshed every  $b$  tuples, therefore the choice of  $b$  is governed by the latency requirements of the user or application. The algorithm returns a list of item types that are expected to occur within the current instance of the sliding window with a relative frequency of at least  $1/m$ , where  $m$  is a user-defined threshold parameter.

Repeat:

1. For each element  $e$  in the next  $b$  elements:  
If a local counter exists for the type of element  $e$ , increment it,  
Otherwise, create a new local counter for this element type and set it equal to 1
2. Add a summary  $S$  containing the element types of all local counters  $\geq b/m$  to the back of queue  $Q$
3. Delete all local counters
4. For each type named in  $S$ :  
If a global counter exists for this type, increment it,  
Otherwise, create a new global counter for this element type and set it equal to 1
5. If  $sizeOf(Q) > N/b$ :
  - (a) Remove the summary  $S'$  from the front of  $Q$
  - (b) Decrement the global counters for all element types named in  $S'$
  - (c) If a counter is decremented to zero, delete it
  - (d) Output the identity and value of all global counters greater than some threshold  $\tau$

\*This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Let  $d$  be the number of distinct item types in the stream. The worst-case space requirement of our algorithm consists of two parts: the working space needed to create a summary for the current basic window, and the storage space needed for the summaries of the individual basic windows. The working space requires  $\min(b, d)$  local counters of size  $\log b$ . For storage, there are  $mN/b$  summaries, each requiring  $\log d$  bits. There are also at most  $mN/b$  global counters of size  $\log(N/b)$ . This gives a total space bound of  $O(\min(b, d) \log b + \frac{mN}{b} \log d + \frac{mN}{b} \log \frac{N}{b})$ . The time complexity is  $O(b)$  for each pass through the outer loop. Since each pass consumes  $b$  arriving elements, this gives  $O(1)$  amortized time per element.

### 3 Analysis

We now discuss how to transform the counts returned by the algorithm into frequency estimates with error bounds, and investigate how to calculate the required value of  $\tau$  used in step 5(d) of the algorithm. Each time a basic window is filled, the algorithm supplies counts  $f(x_i)$  of the number of basic windows in which item types  $x_i$  exceeded the frequency threshold. Recall that item types  $x_i$  are assumed to conform to a multinomial distribution, where the probability that a given tuple belongs to type  $x_i$  is  $p(x_i)$ .

**Proposition 1.** Consider the random variable  $w$  defined as follows.

$$w = \begin{cases} 1 & \text{if } \text{count}(x_i) \geq b/m \text{ in a basic window} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Then,  $w$  constitutes a Bernoulli variable with the probability of “success”  $B(x_i)$  equal to the probability that type  $x_i$  exceeds the relative frequency threshold  $1/m$  within the basic window, as shown in Equation (2).

$$B(x_i) = \sum_{j=\lceil \frac{b}{m} \rceil}^b \binom{b}{j} p(x_i)^j (1 - p(x_i))^{b-j} \quad (2)$$

**Corollary 1.** Since the probability of type  $x_i$  exceeding the frequency threshold  $1/m$  in any one basic window is independent of its probability of exceeding the frequency threshold in any other basic window, the sum of  $n$  Bernoulli variables as defined in Equation (1) is a binomial variable with parameters  $n$  and  $B(x_i)$ .

The frequencies  $f(x_i)$  output by our algorithm may be used to calculate observed relative frequencies  $\hat{B}(x_i)$  that item types  $x_i$  exceed the relative frequency threshold in a basic window:

$$\hat{B}(x_i) = f(x_i)/n. \quad (3)$$

These values can then be substituted in Equation (2) in order to obtain values for  $\hat{p}(x_i)$ , the expected relative frequency of item  $x_i$ .

Consider a sample of  $n$  items from a binomial distribution and an observed frequency of  $f$ . The following is Hoeffding’s bound on the deviation of the observed frequency from the true frequency  $p$  [2].

$$\mathbf{P} \left\{ \frac{f}{n} - p \geq \Delta \right\} \leq e^{-2n\Delta^2} \quad (4)$$

The frequencies returned by our algorithm  $\{f(x_1), \dots, f(x_d)\}$  follow a multinomial distribution with parameters  $n, B(x_1), B(x_2), \dots, B(x_d)$ . Therefore, the marginal distribution for  $f(x_i)$  (and hence  $\hat{B}(x_i)$ ) follows a binomial distribution. Now,  $B(x_i)$  is a binomial random variable—by Corollary 1,  $f(x_i)$  is a binomial random variable, and  $B(x_i)$  is simply a normalized form of  $f(x_i)$ . Using the Hoeffding bound along with a symmetry argument gives the following.

$$\mathbf{P} \left\{ (\hat{B}(x_i) - \Delta) \leq B(x_i) \leq (\hat{B}(x_i) + \Delta) \right\} > 1 - 2e^{-2n\Delta^2} \quad (5)$$

The right-hand side is the confidence level, so by setting it equal to the desired confidence (e.g. 0.95) we can solve for  $\Delta$  (note that  $n$  is fixed by the choice of  $b$ ). Because  $B(x_i)$  in Equation (2) increases monotonically with  $p(x_i)$ , we can find lower and upper bounds for  $p(x_i)$  by numerically computing solutions to Equation (2) for the points  $B(x_i) = (\hat{B}(x_i) - \Delta)$  and  $B(x_i) = (\hat{B}(x_i) + \Delta)$ , respectively.

Finally, we comment on the required value of  $\tau$  used in step 5(d) of the algorithm. Assume that  $1/m$  is the desired threshold. Then, the value for  $\tau$  should be the expected value for  $B(x_i)$  when  $p(x_i)=1/m$ , which can be calculated by substituting  $1/m$  for  $p(x_i)$  in Equation (2). This value for  $\tau$  gives the most likely list of item types that have a relative frequency over  $1/m$ ; see [1] for more details.

### References

- [1] L. Golab, D. DeHaan, A. López-Ortiz, and E. D. Demaine. Finding frequent items in sliding windows with multinomially-distributed item frequencies. University of Waterloo Tech. Report CS-2004-06. Available at <http://db.uwaterloo.ca/~lgolab/cs-2004-06.pdf>.
- [2] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.
- [3] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. *VLDB’02*, pp. 358–369.