Mining Frequent Items in a Stream Using Flexible Windows

Toon Calders
TU Eindhoven
The Netherlands

Nele Dexters   Bart Goethals
University of Antwerp
Belgium

Abstract

We study the problem of finding frequent items in a continuous stream of itemsets. A new frequency measure is introduced, based on a flexible window length. For a given item, its current frequency in the stream is defined as the maximal frequency over all windows from any point in the past until the current state. We study the properties of the new measure, and propose an incremental algorithm that allows to produce the current frequency of an item immediately at any time. It is shown experimentally that the memory requirements of the algorithm are extremely small for many different realistic data distributions.

1 Introduction

Mining frequent items over a stream of itemsets presents interesting new challenges over traditional mining in static databases. It is assumed that the stream can be scanned only once, and hence if an item is passed, it can not be revisited, unless it is stored in main memory. Storing large parts of the stream, however, is not possible because the amount of data passing by is typically huge.

Previous research on mining frequent item(set)s over data streams compute frequencies within one or more windows of a fixed size or with a decay factor. In many applications, however, it is not possible to fix a window length or a decay factor that is most appropriate for every item at every timepoint in an evolving stream. For example, consider a large retail chain of which sales can be considered as a stream. Then, in order to find frequent items to do market basket analysis, it is very difficult to choose in which period of the collected data you are particularly interested. For many products, the amount sold depends highly on the period of the year. E.g., in summer time, sales of ice cream increase and during the world cup, sales of beer increase. Such seasonal behavior of a specific item can only be discovered when choosing the correct window size for that item. This size, however, can hide a similar behavior of other items in another window. Therefore, we propose a new frequency measure that, for each item, chooses the window in which it is best represented. More specifically, we define the current frequency of an item as its maximum frequency over all possible windows from the past until the current state. Obviously, the most recent item in the stream always has a frequency of 100% in its window of size 1. This disadvantage can be resolved by only considering windows larger than a given minimum window length. When the stream evolves, the length of the window containing the highest frequency for a given item can grow and shrink continuously. We show some important properties of this behavior.

In our approach, on every timestamp, a new itemset arrives in the stream. We present an incremental algorithm that maintains a small summary of relevant information of the history of the stream that allows to produce the current frequency of a specific item in the stream immediately at any time. That is, when a new itemset arrives, the summary is updated, and when at a certain point in time, the current frequency of an item is required, the result can be obtained instantly from the summary. The structure of the summary is based on some critical observations about the windows with the maximal frequency. In short, many points in the stream can never become the starting point of a maximal window, no matter what the continuation of the stream will be. The summary will thus consist of some statistics about the few points in the stream that are still candidate starting points of a maximal window. These important points in the stream will be called the *borders*.

Critical for the usefulness of the technique are the memory requirements of the summary that needs to be maintained in memory. We show experimentally that, even though in worst case the summary depends on the length of the stream, for realistic data distributions its size is extremely small. Obviously, this property is highly desirable as it allows for an efficient and effective computation of our new measure. Also note that our approach allows exact information as compared to many approximations considered in other works.

The organization of the paper is as follows. In Section 2, we start with a discussion on related work in stream mining for frequent item(set)s. Then, in Section 3, our new measure is defined and the central problem statement is formally introduced. Section 4 gives the theoretical results and the main theorem, on which the incremental algorithm in Section 5 is based. Section 6 contains experiments that show that the memory requirements for the algorithm are extremely small for many real-life data distributions. Section 7 concludes.

2 Related Work

In the next sections we will discuss a new frequency measure for finding frequent items in a stream. This frequency measure is quite different from the ones typically found in the literature on stream mining.

In the *sliding window* model, the importance of the data contained in the sliding window of fixed length is emphasized and the goal is to discover recent usage trends [1, 4, 7, 9, 12]. Thus, in this model the focus is on the most recent items in a window of fixed length. The size of the window is fixed at the start of the analysis and the content of the window changes when the stream (and time) continues. Two different points of view are considered: the count-based view, in which the number of transactions in the window is fixed despite the generating speed of the transactions, and the time-based windows, in which the number of time units in the window is fixed, causing the number of transactions to be different. When a transaction leaves the sliding window, it is eliminated and does not contribute to the frequency measure anymore.

In the *time-fading* model, the sensitivity of time is emphasized in the sense that recent transactions get a higher weight as compared to earlier transactions. This is achieved by introducing a decay mechanism [10]. In addition to this mechanism, a tilted-time window can be introduced [5, 6]. This technique reflects the alteration of the time scales of the windows as time goes by. Recent data is mined at a fine granularity while long-term data is mined at a coarse granularity.

In the *landmark* model, a particular time period is fixed, from the landmark designating the start of the system untill the current time [8, 9, 15]. The analysis of the stream is performed for only the part of the stream between the landmark and the current time instance. A major disadvantage of this method is that the size of the window varies; it starts with size zero and grows until the next occurrence of the landmark, at which point it is reset to zero.

3 A New Frequency Measure in Stream Mining

In this section, we define our new frequency measure for streams and we formally introduce the problem. Throughout the paper, we assume all items occurring in the stream come from a finite set of items $\mathcal{I}$.

3.1 Streams and Max-Frequency

In general, a *stream* $\langle I_1 \ I_2 \ \dots \ I_n \rangle$ is a sequence of itemsets, denoted $\mathbb{S}$, where n is the *length* of the stream, denoted $|\mathbb{S}|$. The stream of length 3, having on the first timestamp the singleton $\{a\}$, on the second timestamp the singleton $\{b\}$ and on the third timestamp the singleton $\{c\}$, is denoted $\langle a \ b \ c \rangle$, and the stream of length 3, with on the first timestamp the set $\{a, b\}$, on the second timestamp the singleton $\{c\}$ and on the third timestamp the set $\{a, d, f\}$, is denoted $\langle ab \ c \ adf \rangle$.

The number of sets in a stream $\mathbb{S}$ that contain item i will be denoted $count(i, \mathbb{S})$. For example, $count(a, \langle a \ b \ c \rangle)$ is 1, since a occurs in exactly one set. For the other stream we have $count(a, \langle ab \ c \ adf \rangle) = 2$.

The *frequency of i in* $\mathbb{S}$, is defined as

$$freq(i, \mathbb{S}) := \frac{count(i, \mathbb{S})}{|\mathbb{S}|} \quad .$$

For example, $freq(a, \langle a \ b \ c \rangle) = 1/3$ and $freq(a, \langle ab \ c \ adf \rangle) = 2/3$.

Let $\mathbb{S}_1$ be $\langle I_1^1 \ldots I_{n_1}^1 \rangle$, $\mathbb{S}_2$ is $\langle I_1^2 \ldots I_{n_2}^2 \rangle$, $\ldots$ and $\mathbb{S}_m$ is $\langle I_1^m \ldots I_{n_m}^m \rangle$. The *concatenation* of $\mathbb{S}_1, \ldots, \mathbb{S}_m$, denoted $\mathbb{S}_1 \cdot \mathbb{S}_2 \cdot \ldots \cdot \mathbb{S}_m$, is

$$\langle\ I_1^1\ \ldots\ I_{n_1}^1\ I_1^2\ \ldots\ I_{n_2}^2 \ldots\ I_1^m\ \ldots\ I_{n_m}^m\ \rangle\ .$$

Let $\mathbb{S} = \langle I_1\ I_2\ \ldots\ I_n \rangle$. Then, $\mathbb{S}[s,t]$ denotes the *sub-stream* or *window* $\langle I_s\ I_{s+1}\ \ldots\ I_t \rangle$. The sub-stream of $\mathbb{S}$ consisting of the last k itemsets of $\mathbb{S}$, denoted $last(k, \mathbb{S})$, is

$$last(k, \mathbb{S}) := \mathbb{S}\big[|\mathbb{S}| - k + 1, |\mathbb{S}|\big]\ .$$

We are now ready to define our new frequency measure:

Definition 1 *The* max-frequency $mfreq(i, \mathbb{S})$ *of an item i in a stream $\mathbb{S}$ is defined as the maximum of the frequencies of i over all windows extending from the end of the stream; that is:*

$$mfreq(i, \mathbb{S})\ :=\ \max_{k=1\ldots|\mathbb{S}|}(freq(i, last(k, \mathbb{S})))\ .$$

The longest window in which the max-frequency is reached, is called the maximal window for i in $\mathbb{S}$*, and its starting point is denoted $maxwin(i, \mathbb{S})$. That is, $maxwin(i, \mathbb{S})$ is the smallest index such that*

$$mfreq(i, \mathbb{S})\ =\ freq(i, \mathbb{S}\big[maxwin(i, \mathbb{S}), |\mathbb{S}|\big])\ .$$

The measure $mfreq(i, \mathbb{S})$ is used as a new frequency measure for stream mining. For a given item, its current frequency in the stream is defined as the maximal frequency over all evolving windows from a point in the stream until the end. Note that, by definition, the max-frequency of an item a in a stream that ends with an itemset containing a, is always 100%, independently of the overall frequency of a. Hence, even in a stream where a is extremely rare, at some points, the max-frequency will be maximal! This disadvantage of max-frequency, however, can easily be resolved by setting a minimal length for all windows. We did not include this solution in the paper, as it is not fundamental for the developed theory, and it would unnecessarily complicate our explanations.

Example 2 *We focus on target item a.*

$$\begin{aligned}
mfreq(a, \langle a\ b\ a\ a\ a\ b \rangle) &= \max_{k=1\ldots6}(freq(a, last(k, \langle a\ b\ a\ a\ a\ b \rangle))) \\
&= \max\left(\frac{0}{1}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{3}{5}, \frac{4}{6}\right) = \frac{3}{4}\ . \\
mfreq(a, \langle b\ c\ d\ a\ b\ c\ d\ a \rangle) &= \max\left(\frac{1}{1}, \cdots\right) = 1\ . \\
mfreq(a, \langle x\ a\ a\ x\ a\ a\ x \rangle) &= \max\left(\frac{0}{1}, \frac{1}{2}, \frac{2}{3}, \frac{2}{4}, \frac{3}{5}, \frac{4}{6}, \frac{4}{7}\right) = \frac{2}{3}\ .
\end{aligned}$$

3.2 Problem Statement

Notice that a stream is defined as a statical object. In reality, however, a stream is an evolving object. At every timepoint, a new itemset will be inserted at the end of the stream. As such, evolving streams are essentially unbounded, and when processing them, it is to be assumed that only a small part can be kept in memory.

Hence, an evolving stream $\mathbb{S}$ is an unbounded sequence. $\mathbb{S}_t$ will denote the stream $\mathbb{S}$ up to timestamp t; that is, the part of the stream that already passed at timestamp t. In the examples, the streams will grow from left to right. This means that the older itemsets are on the left of the stream, and the newest, or more recent itemsets on the righthand side. For simplicity we assume that the first itemset arrived at timestamp 1, and since then, at every timestamp a new itemset was inserted.

The problem we study in this article is the following: *For an evolving stream* $\mathbb{S}$ *and a fixed item* a*, maintain a small summary of the stream in time, such that, at any timepoint* t*,* $\mathit{mfreq}(a, \mathbb{S}_t)$ *can be produced instantly from the summary.*

More formally, we introduce a summary of a stream $\mathit{summary}(\mathbb{S})$, an update procedure *Update*, and a procedure *Get_mfreq*, such that, if on timestamp $t+1$ an itemset containing item i is added to the stream, $\mathit{Update}(\mathit{summary}(\mathbb{S}_t), i)$ equals $\mathit{summary}(\mathbb{S}_t \cdot \langle i \rangle)$ equals $\mathit{summary}(\mathbb{S}_{t+1})$, and $\mathit{Get\_mfreq}(\mathit{summary}(\mathbb{S}_{t+1}))$ equals $\mathit{mfreq}(a, \mathbb{S}_{t+1})$. Because *Update* has to be executed every time a new itemset arrives, it has to be very efficient, in order to be finished before the next itemset arrives. Similarly, because the stream continuously grows, the summary must be independent of the number of itemsets seen so far, or, at least grow very slowly as the stream evolves. The method we develop will indeed meet these criteria, as the experiments will show.

stream	time	$\mathit{mfreq}(a, \mathbb{S}_t)$
$\langle \mid a \rangle$	1	$\max(\frac{1}{1}) = 1$
$\langle \mid a \quad a \rangle$	2	$\max(\frac{1}{1}, \frac{2}{2}) = \frac{2}{2} = 1$
$\langle \mid a \quad a \quad a \rangle$	3	$\max(\frac{1}{1}, \frac{2}{2}, \frac{3}{3}) = \frac{3}{3} = 1$
$\langle \mid a \quad a \quad a \quad b \rangle$	4	$\max(\frac{0}{1}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}) = \frac{3}{4}$
$\langle \mid a \quad a \quad a \quad b \quad b \rangle$	5	$\max(\frac{0}{1}, \frac{0}{2}, \frac{1}{3}, \frac{2}{4}, \frac{3}{5}) = \frac{3}{5}$
$\langle \mid a \quad a \quad a \quad b \quad b \quad b \rangle$	6	$\max(\frac{0}{1}, \frac{0}{2}, \frac{0}{3}, \frac{1}{4}, \frac{2}{5}, \frac{3}{6}) = \frac{3}{6}$
$\langle \quad a \quad a \quad a \quad b \quad b \quad b \mid a \rangle$	7	$\max(\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{2}{5}, \frac{3}{6}, \frac{4}{7}) = 1$
$\langle \quad a \quad a \quad a \quad b \quad b \quad b \mid a \quad a \rangle$	8	$\max(\frac{1}{1}, \frac{2}{2}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{3}{6}, \frac{4}{7}, \frac{5}{8}) = \frac{2}{2}$
...	...	...

Figure 1: Max-frequency of a stream at every timepoint.

In Fig. 1, the max-frequency has been given for an example evolving stream. The starting point $\mathit{maxwin}(a, \mathbb{S})$ of each maximal window is marked with $|$.

3.3 Minimal Frequency

Not only the maximal frequency of an item can be of interest for an analyst, also the *minimal frequency* can sometimes reveal interesting knowledge. Inspired by the max-frequency measure, we can also define the min-frequency of item a as the minimum of the frequency of a over all windows extending from the end of the stream.

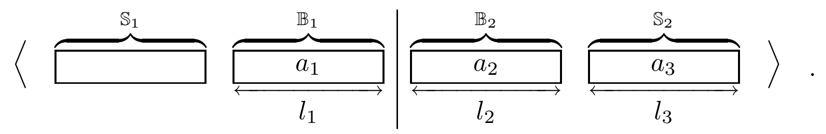
Notice, however, that maintaining and finding the minimal frequency does not pose a new challenge over the maximal frequency. Indeed, consider $\overline{a}$, the negation of a, as if it is a new item. Then it is easy to see that at any timepoint the maximal frequency of $\overline{a}$ and the minimal frequency of a add up to 1. Therefore, in order to find the minimal frequency of a, we can in fact monitor the imaginary item $\overline{a}$, and when we need the minimal frequency of a, it is easily found as $1 - \mathit{mfreq}(\overline{a}, \mathbb{S})$.

4 Properties of Max-Frequency

In this section we show some properties of max-frequency that are crucial for the incremental algorithm that maintains the summary of the stream. Obviously, checking all possible windows to find the maximal one is infeasible.Fortunately, not every point in the stream needs to be checked. The theoretical results from this section show exactly which points need to be inspected. These points will be called the *borders*.The summary of the stream will consist exactly of the recording of these borders, and the frequency of the target item up to the most recent timepoint.

Theorem 1 *Consider a stream* $\mathbb{S} := \mathbb{S}_1 \cdot \mathbb{B}_1 \cdot \mathbb{B}_2 \cdot \mathbb{S}_2$*. If* $\mathbb{B}_2 \cdot \mathbb{S}_2$ *is the maximal window for* a *in* $\mathbb{S}$*, then* $\mathit{freq}(a, \mathbb{B}_1) < \mathit{freq}(a, \mathbb{B}_2)$

Proof If $\mathbb{B}_2 \cdot \mathbb{S}_2$ is the maximal window for a in $\mathbb{S}$, then this implies that the frequency of a in $\mathbb{B}_2 \cdot \mathbb{S}_2$ is strictly higher than in $\mathbb{B}_1 \cdot \mathbb{B}_2 \cdot \mathbb{S}_2$ and at least as high as in $\mathbb{S}_2$ (remember that in the case of multiple windows with maximal frequency, the largest one is selected). Let now $l_1 = |\mathbb{B}_1|$, $l_2 = |\mathbb{B}_2|$, and $l_3 = |\mathbb{S}_2|$, and let $a_1 = \mathit{count}(a, \mathbb{B}_1)$, $a_2 = \mathit{count}(a, \mathbb{B}_2)$, and $a_3 = \mathit{count}(a, \mathbb{S}_2)$, as depicted in:



Then, the conditions on the frequency translate into:

$$\frac{a_2 + a_3}{l_2 + l_3} > \frac{a_1 + a_2 + a_3}{l_1 + l_2 + l_3} \quad \text{and} \quad \frac{a_2 + a_3}{l_2 + l_3} \geq \frac{a_3}{l_3}.$$

From these conditions, it can be derived that

$$\mathit{freq}(a, \mathbb{B}_1) = \frac{a_1}{l_1} < \frac{a_2}{l_2} = \mathit{freq}(a, \mathbb{B}_2). \qquad \square$$

Based on this theorem, it is possible to give an exact characterization of which points in $\mathbb{S}_t$ can potentially become the starting point of the maximal window in the future, after new items have been added. First, we formally define the important notion of a *border*. Intuitively, a border is a point in the stream that can still become the starting point of the maximal window.

Definition 3 *The position q in $\mathbb{S}$ is called a* border for a in $\mathbb{S}$ *if there exists another stream $\mathbb{B}$ such that $q = \mathit{maxwin}(a, \mathbb{S} \cdot \mathbb{B})$.*

Corollary 2 *Let $\mathbb{S}$ be a stream, and let $q = 1$. Position q is a border for item a if and only if the first itemset in the stream contains the target item a.*

Let $\mathbb{S}$ be a stream, and let $2 \leq q \leq |\mathbb{S}|$. Position q is a border for item a in $\mathbb{S}$ if and only if for all indices j, k with $1 \leq j < q$ and $q \leq k \leq |\mathbb{S}|$, it holds that $\mathit{freq}(a, \mathbb{S}[j, q-1]) < \mathit{freq}(a, \mathbb{S}[q, k])$.

Proof Only if: Suppose that there exist indices j and k, and a stream $\mathbb{B}$ such that $\mathit{freq}(a, \mathbb{S}[j, q-1]) \geq \mathit{freq}(a, \mathbb{S}[q, k])$, and $q = \mathit{maxwin}(a, \mathbb{S} \cdot \mathbb{B})$. This situation is in contradiction with Theorem 1: split the stream $\mathbb{S} \cdot \mathbb{B}$ as $(\mathbb{S}[1, j-1]) \cdot (\mathbb{S}[j, q-1]) \cdot (\mathbb{S}[q, k]) \cdot (\mathbb{S}[k+1, |\mathbb{S}|] \cdot \mathbb{B})$. In this stream, $(\mathbb{S}[q, |\mathbb{S}|]) \cdot \mathbb{B}$ is the maximal window, while $\mathit{freq}(a, \mathbb{S}[j, q-1]) \geq \mathit{freq}(a, \mathbb{S}[q, k])$.

If: We need to show that there exists a continuation $\mathbb{S} \cdot \mathbb{S}'$ of stream $\mathbb{S}$ in which q is the starting point of the maximal window. We consider two cases: either q is the rightmost border in $\mathbb{S}$, or not. If q is the rightmost border, then q is the maximal border in $\mathbb{S}$, because for any other border $p < q$, $\mathit{freq}(a, \mathbb{S}[p, q-1]) < \mathit{freq}(a, \mathbb{S}[q, |\mathbb{S}|])$ which implies $\mathit{freq}(a, \mathbb{S}[p, |\mathbb{S}|]) < \mathit{freq}(a, \mathbb{S}[q, |\mathbb{S}|])$.

In the other case, let q' be the leftmost border in $\mathbb{S}$ that is on the right of q. That is, $q' > q$ is a border, and there is no other border r with $q' > r > q$. We are now going to construct a stream $\mathbb{S}'$ such that, if we append $\mathbb{S}'$ to $\mathbb{S}$, the frequency of a from q until the end of the extended stream becomes the same as the frequency of a in the window from q to $q' - 1$:

$$\mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, |\mathbb{S} \cdot \mathbb{S}'|]) = \mathit{freq}(a, \mathbb{S}[q, q'-1]) \ .$$

Let y be the length of the extension $\mathbb{S}'$ we are going to construct, and let x be the number of occurrences of the target item a. The condition thus becomes:

$$\mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, |\mathbb{S} \cdot \mathbb{S}'|]) = \frac{\mathit{count}(a, \mathbb{S}[q', |\mathbb{S}|]) + x}{|\mathbb{S}| - q' + 1 + y} = \mathit{freq}(a, \mathbb{S}[q, q'-1]) \ .$$

Notice that for any solution x, y of the above equality, always $x \leq y$, because

$$\mathit{freq}(a, \mathbb{S}[q, q'-1]) < \mathit{freq}(a, \mathbb{S}[q', |\mathbb{S}|]) = \frac{\mathit{count}(a, \mathbb{S}[q', |\mathbb{S}|])}{|\mathbb{S}| - q' + 1} \ ,$$

since q' is a border, and thus every block before q' must have frequency lower than any block after q' (according to the only if part which is already proven above).

$\mathbb{S}'$ is now the following stream:

$$\langle \overbrace{a\ a\ \cdots\ a}^{x\times} \overbrace{\emptyset\ \emptyset\ \cdots\ \emptyset}^{y-x\times} \rangle \ .$$

All what is left to show is that q is, indeed, the maximal border in $\mathbb{S} \cdot \mathbb{S}'$. Obviously, because all target letters in $\mathbb{S}'$ occur in the beginning of the stream, the maximal frequency of a in $\mathbb{S}'$ is x/y, which is always smaller than $\mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, |\mathbb{S} \cdot \mathbb{S}'|])$. This relation can be seen as follows: $(\mathbb{S} \cdot \mathbb{S}')[q, |\mathbb{S} \cdot \mathbb{S}'|]$ can be split into three parts: $\mathbb{S}[q, q'-1]$, which has a frequency of a equal to that of the whole, $\mathbb{S}[q', |\mathbb{S}|]$, which has a frequency of a strictly larger than of the first part (q' is a border in $\mathbb{S}$, the first part a before-block, and the second part an after-block of q' in $\mathbb{S}$), and the third part $\mathbb{S}'$ that consequently needs to have a frequency of a lower than that of the whole.

Furthermore, for any border q'' after q in $\mathbb{S}$,

$$\mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, q''-1]) \geq \mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, q'-1]) = \mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, |\mathbb{S} \cdot \mathbb{S}'|]) \ ,$$

Hence,

$$\mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q, q''-1]) \geq \mathit{freq}(a, (\mathbb{S} \cdot \mathbb{S}')[q'', |\mathbb{S} \cdot \mathbb{S}'|]) \ ,$$

and thus, there is a before-block for q'' that has a larger frequency of a's than an after-block. Therefore, q'' cannot be a border. Furthermore, for all borders p before q in $\mathbb{S}$,

$$\mathit{freq}(a, \mathbb{S}[p, q-1]) < \mathit{freq}(a, \mathbb{S}[q, q'-1]) = \mathit{freq}(a, \mathbb{S}[q, |\mathbb{S} \cdot \mathbb{S}'|]) \ ,$$

because $\mathbb{S}[p, q-1]$ is a before-block and $\mathbb{S}[q, q'-1]$ an after block of q in $\mathbb{S}$. Therefore, $\mathbb{S}[q, |\mathbb{S} \cdot \mathbb{S}'|]$ is the maximal window for a in $\mathbb{S} \cdot \mathbb{S}'$. □

Example 4 *Assume we have the following stream $\mathbb{S}_{27}$:*

$$\langle\ \overset{4/9}{\boxed{a\ a\ a\ b\ b\ b\ a\ b\ b}}\ \Big|\ \overset{4/10}{\boxed{a\ b\ a\ b\ a\ b\ a\ b\ b\ b}}\quad b\quad \overset{2/3}{\boxed{a\ a\ b}}\ \Big|\ \overset{1/2}{\boxed{a\ b}}\quad b\ a\ \rangle$$

In the stream, two positions have been marked with $|$. Both these points do not meet the criteria given in Corollary 2 to be a border. Indeed, for both positions, a block before and after it is indicated such that the frequency in the before-block is higher than in the after-block.
In this stream, the only positions that meet the requirement are indicated by vertical bars: $\langle |a\ a\ a\ b\ b\ b\ a\ b\ b\ a\ b\ a\ b\ a\ b\ a\ b\ b\ b\ b\ |a\ a\ b\ a\ b\ b\ |a\rangle$.
Consider now the second border (at position 21). The frequency of a between this border and the next border (position 27) is 0.5. According to the proof of Corollary 2, we can compute a continuation of the stream such that position 21 becomes the maximal border as follows: first we have to find integers x and y such that

$$0.5 = \frac{\mathit{count}(a, \mathbb{S}[27, |\mathbb{S}_{27}|]) + x}{|\mathbb{S}_{27}| - 27 + 1 + y} = \frac{1+x}{1+y} \ .$$

This condition is satisfied with $x = 0$ and $y = 1$. This means that we need to add a stream of length 1 starting with 0 times the target; that is, e.g., $\langle b\rangle$. Note that this extension of the stream is constructed in such a way that the frequency of a from position 21 until the end of the stream becomes equal to the frequency of a in the block between the border at position 21 and the next border at position 27. Therefore, position 21 is indeed a border in $\mathbb{S}_{27}\cdot\langle b\rangle$.

The following simple facts have an important role in the algorithm that is developed in the next section.

Corollary 3 *A border is always located at the timestamp of an itemset that contains the target item. If p is not a border in $\mathbb{S}$, then it can never be a border in any extension $\mathbb{S}\cdot\mathbb{B}$.*

Proof Let p be a border. Suppose there is no target item at position p. Then p does not satisfy the condition of Corollary 2, because $\mathit{freq}(a, \mathbb{S}[p, p]) = 0 \leq \mathit{freq}(a, \mathbb{S}[p-1, p-1])$.

By definition, a position p is a border if there exists an extension of the stream such that p is the starting position of the maximal window in this extension. Hence, if p is not a border in $\mathbb{S}$, then neither it is in $\mathbb{S}\cdot\mathbb{S}'$, as every extension of $\mathbb{S}\cdot\mathbb{S}'$ is also an extension of $\mathbb{S}$. □

5 Algorithm

Based on the theorems of Section 4, we now present an incremental algorithm to maintain a summary.

5.1 The Summary

The summary for an item a in the stream $\mathbb{S}_t$ is the array that contains a pair $(p, x/y)$ for every border at position p, with x the number of occurrences of a since p, i.e., $\mathit{count}(a, \mathbb{S}_t[p, t])$, and y the length of the block from p until the end of the stream $\mathbb{S}_t$, i.e., $t - p + 1$. The output of the algorithm in the case of r borders is written as an array of the form $[(p_1, x_1/y_1), \cdots, (p_r, x_r/y_r)]$, visualized by

$$T_t(a) = \begin{array}{|c|c|c|}\hline p_1 & \cdots & p_r \\ \hline x_1/y_1 & \cdots & x_r/y_r \\ \hline\end{array} \ .$$

This array is in fact $\mathit{summary}(\mathbb{S}_t)$ for item a, and is denoted by $T_t(a)$. If the target item is clear from the context, we use the abbreviation T_t. In this array, the border positions are ordered from old to most recent, reflecting in $p_1 < \cdots < p_r$. The corresponding frequencies must follow the same ordering $x_1/y_1 < \cdots < x_r/y_r$. Indeed, consider two consecutive borders p_i and p_{i+1}. Suppose for the sake of contradiction, that $x_i/y_i \geq x_{i+1}/y_{i+1}$. From this, it follows that

$$\mathit{freq}(a, \mathbb{S}[p_i, p_{i+1} - 1]) = \frac{x_i - x_{i+1}}{y_i - y_{i+1}} \geq \frac{x_i}{y_i} = \mathit{freq}(a, \mathbb{S}[p_{i+1}, |\mathbb{S}|]) \ .$$

According to Theorem 1 this implies that p_{i+1} cannot be a border because the frequency of a in a before-block is at least the frequency of a in an after-block for p_{i+1}. *Notice that this implies that the current frequency can always be obtained immediately from the summary; the most recent entry in the summary is always the largest and thus holds the current max-frequency.*

In every next step, the algorithm adjusts the stored values in the array based on the newly entered itemset in the stream. Hence, at every step, we need to test for every border of $\mathbb{S}_t$ if it is still a border in $\mathbb{S}_{t+1}$. Thus, we need to check if the frequency in all before-blocks is still smaller than the frequency in all after-blocks. *However, adding a new itemset to the stream does not introduce a new before-block, and only one after-block!* Thereofore, only one test has to be performed for every border of $\mathbb{S}_t$: the before-block with the highest frequency of a has to be compared to the new after-block. The frequency of the new after-block for border p_i can easily be obtained from the pair $(p_i, x_i/y_i)$ in the summary of $\mathbb{S}_t$: if the new itemset is a non-target itemset (an itemset that does not contain the target), the frequency of a in the new after-block is $x_i/(y_i+1)$. *Notice that, as the before-block never changes when a new itemset enters the stream, and the insertion of the target itemset (a set that contains the target item a)only results in an increased frequency in the new after-block, the addition of a target itemset will never result in the removal of borders.*

Based on the observation that in any summary the borders must be in order w.r.t. the frequency of a, it is not too hard to see that the before-block with the maximal frequency is exactly the block $\mathbb{S}_t[p_{i-1}, p_i - 1]$. Using a similar reasoning as above, it follows that p_i is still a border for $\mathbb{S}_{t+1}$ if and only if the updated frequency $x_i/(y_i+1)$ is still larger than the updated frequency $x_{i-1}/(y_i+1)$ of p_{i-1}. To summarize, we have the following properties:

- The frequencies in the summary are always increasing.
- When a target itemset is added to the stream, all borders remain borders. The frequencies of the borders can be updated by incrementing all nominators and denominators by 1. A new entry with the current timestamp and frequency 1/1 can be added, unless the last entry also has 100% frequency.
- If a non-target itemset enters the stream, the frequencies of the borders can be updated by adding 1 to the denominators of the frequencies. All former borders for which, after the update, the frequency is no longer larger than in the previous entry, are no longer a border.

5.2 The Algorithm

Before the first target itemset enters the stream, the array will remain empty. The pseudo-code of the algorithm to create T_{t+1}, based on T_t and the itemset I that enters the stream at time $t+1$ is given in Algorithm 1. In short, when a new itemset I enters the stream, the frequencies are updated by increasing the nominators if I contains the target item, and always increasing the denominators. If the itemset I contains the target item, a new border will be added

Algorithm 1 *Update*(T_t, I) for target item a on time $t+1$

```
Require: $T_t = summary(\mathbb{S}_t) = [(p_1, x_1/y_1), \ldots, (p_r, x_r/y_r)]$
Ensure: $T_{t+1} = summary(\mathbb{S}_{t+1}) = summary(\mathbb{S}_t \cdot \langle i \rangle)$
 1: Set $T_{t+1} := [\ ]$
 2: if ($T_t$ is empty) then
 3:   if (target item $a \in I$) then
 4:     $T_{t+1} := [(t+1, 1/1)]$
 5: else
 6:   if (target item $a \in I$) then
 7:     for $1 \leq j \leq r$ do
 8:       $T_{t+1} := T_{t+1} + \big(p_j, (x_j+1)/(y_j+1)\big)$
 9:     if $x_r \neq y_r$ then
10:       $T_{t+1} := T_{t+1} + \big(t+1, 1/1\big)$
11:   else
12:     $high := 0$
13:     for all $j := 1 \ldots r$ do
14:       if $(x_j)/(y_j+1) > high$ then
15:         $T_{t+1} := T_{t+1} + \big(p_j, (x_j)/(y_j+1)\big)$
16:         $high := (x_j)/(y_j+1)$
```

only if the frequency of the last block in T_t was not equal to 1. Furthermore, we have to take into account that some of the borders of $\mathbb{S}_t$ might no longer be borders in $\mathbb{S}_{t+1}$. This can only happen if the itemset that enters the stream is a non-target itemset (does not contain the target item), and is tested in lines 12-16: the frequencies have to increase for increasing positions of the borders.

An execution of the algorithm is explained in detail for the following stream of length 27 $\langle b\ a\ a\ a\ b\ a\ a\ b\ a\ b\ b\ a\ a\ a\ a\ b\ a \rangle$ and target item a. For each timestamp, the output of the algorithm is given in Figure 2.

In this example, some interesting things happen. First, the stream starts with non-target itemset $\{b\}$. Therefore, $Update(T_0, \{b\}) = Update([\ ], \{b\})$ on timestamp 1 remains empty, i.e., $T_1 = [\ ]$. The algorithm in fact really starts at timestamp 2. At this moment, $Update([\ ], \{a\})$ results in $T_2 = [(2, 1/1)]$, corresponding to the stream $\langle b\ |a \rangle$ with a border at position 2. On timestamp 8, another interesting fact happens. $T_7 = [(2, 5/6), (6, (2/2))]$, corresponding with the stream $\langle b\ |a\ a\ a\ b\ |a\ a \rangle$. $Update(T_7, \{b\})$ will yield $T_8 = [(2, 5/7)]$, and not $[(2, 5/7), (6, 2/3)]$, because the frequency decreases from the border at position 2 to the border at position 6, and hence, we can conclude that position 6 is no longer a border. This is reflected in $\langle b\ |a\ a\ a\ b\ a\ a\ b \rangle$.

6 Experiments

From the description of the algorithm it is clear that the update procedure is very efficient, given the summaries remain small. Producing the current frequency

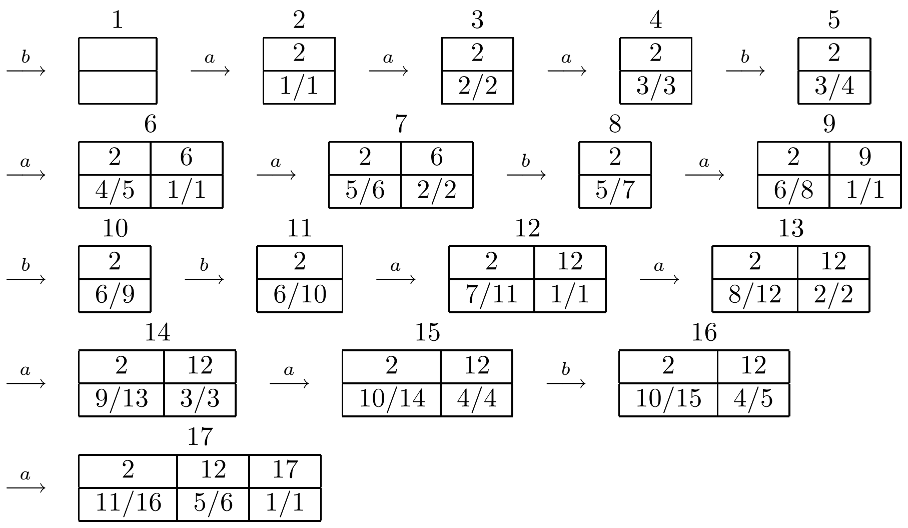


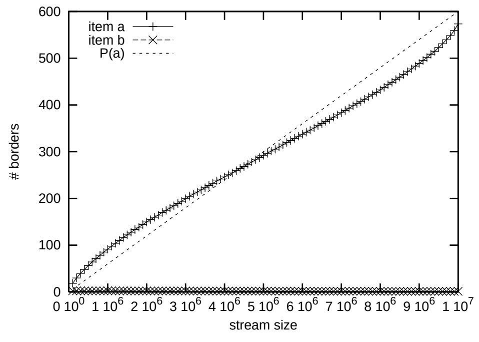
Figure 2: Evolution of the summary for the following stream of length 17: $\langle b\ a\ a\ a\ b\ a\ b\ b\ a\ a\ a\ a\ b\ a \rangle$.

of the target item is obviously very efficient, as it amounts to a simple lookup in the most recent entry. Hence, the complete approach will be feasible if and only if the summaries remain small. Therefore, for different streams, we have recorded the size of the summary. The results are reported in Figure 3.

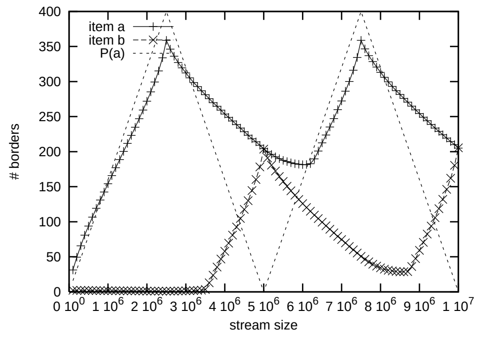
The streams that we consider are over the itemsets that contain items a and b, and have length 10^7. After every 10 000 itemsets, the size of the summary for the items a and b is reported. The streams are randomly generated using different distributions. The probability of having an itemset that contains item a in the stream is given by the line $P(a)$. Thus, for the purely random distribution (Figure 3c), the probability of item a is always 1/2, independent of the time. The probability of b is 1 minus the probability of a. The figures report the average over 100 streams, generated with the indicated distributions. In general, we can conclude that the size of the summary is extremely small w.r.t. the size of the stream. If the probability of the target item increases, also the size of the summary will increase, when the probability decreases, the summary will shrink. This is easily explained by the entries in the summary that need to have increasing frequency.

7 Conclusion and Future Work

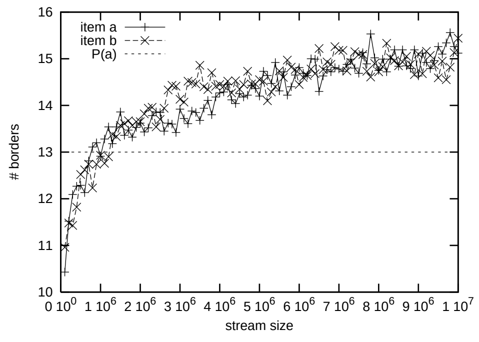
We presented a new frequency measure for items in streams that does not rely on a fixed window length or a time-decaying factor. Based on the properties of the measure, an algorithm to compute it was shown. An experimental evaluation supported the claim that the new measure can be computed from a



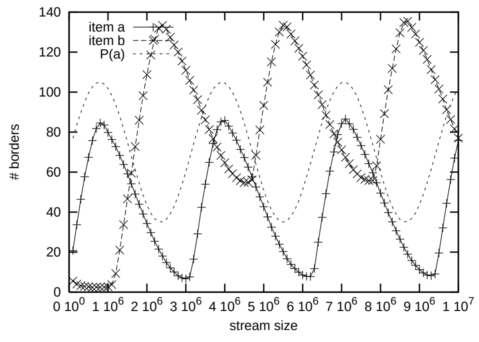
(a) linear distribution



(b) twin peaks distribution



(c) random distribution



(d) sinus distribution

Figure 3: Size of the summaries for two items a and b

summary with extremely small memory requirements, that can be maintained and updated efficiently.

In the future, we will look at the same topic, but try to mine for frequent itemsets instead of items, based on this new frequency measure.

References

[1] Ruoming J. and Agrawal G.: An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In *Proc. 5th IEEE Int. Conf. on Data Mining (ICDM'05)*, pp 210–217. (2005)

[2] Agrawal R., Imielinski T. and Swami A.: Mining Association Rules between Sets of Items in Large Databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp 207–216. (1993)

[3] Cormode G. and Muthukrishnan S.: What's Hot and What's Not: Tracking Most Frequent Items Dynamically. In *Proc. PODS Int. Conf. Principles of Database Systems* (2003)

[4] Demaine E.D., Lopez-Ortiz A. and Munro, J.I.: Frequency Estimation of Internet Packet Streams with Limited Space. In *Proc. of the 10th Annual European Symposium on Algorithms*, pp 348–360. (2002)

[5] Giannella C., Han J., Robertson E. and Liu C.: Mining Frequent Itemsets Over Arbitrary Time Intervals in Data Streams. In *Technical Report TR587*, Indiana University, Bloomington. (2003)

[6] Giannella C., Han J., Pei J., Yan X. and Yu P.S.: Mining Frequent Patterns In Data Streams at Multiple Time Granularities. In H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (eds), *Next Generation Data Mining*, pp 191–212. (2003)

[7] Golab L., DeHaan D.,Demaine E.D., Lopez-Ortiz A. and Munro J.I.: Identifying Frequent Items in Sliding Windows over On-Line Packet Streams. In *Proc. of the 1st ACM SIGCOMM Internet Measurement Conference*, pp 173–178. (2003)

[8] Jin R. and Agrawal G.: An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)*, pp 210–217. (2005)

[9] Karp, R. M., Papadimitriou, C. H. and Shenker, S.: A Simple Algorithm for Finding Frequent Elements in Streams and Bags. In *ACM Trans. on Database Systems* 28, pp 51–55. (2003)

[10] Lee, D. and Lee, W.: Finding Maximal Frequent Itemsets over Online Data Streams Adaptively. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)*, pp 266–273. (2005)

[11] Lee, L.K. and Ting, H.F.: A Simpler and More Efficient Deterministic Scheme for Finding Frequent Items over Sliding Windows. In *Proc. PODS Int. Conf. Principles of Database Systems.* (2006)

[12] Lin C.-H., Chiu D.-Y., Wu Y.-H. and Chen A.L.P.: Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window. In *Proc. SIAM International Conference on Data Mining.* (2005)

[13] Misra J. and Gries, D.: Finding Repeated Elements. In *Science of Computer Programming* , 2(2), pp 143–152. (1982)

[14] Chi-Wing Wong R. and Wai-Chee Fu A.: Mining Top-K Frequent itemsets from Data Streams. In *Data Mining and Knowledge Discovery.* To appear. (2006)

[15] Yu J.X., Chong Z., Lu H. and Zhou A.: False Positive or False Negative: Mining Frequent Items from High Speed Transactional Data Streams. In *Proc. of the 30th International Conference on Very Large Databases*, pp 204–215. (2004)