FMU: Fast Mining of Probabilistic Frequent Itemsets in Uncertain Data Streams

Reza Akbarinia, Florent Masseglia

FMU: Fast Mining of Probabilistic Frequent Itemsets in Uncertain Data Streams

REZA AKBARINIA AND FLORENT MASSEGLIA

INRIA and LIRMM, Montpellier, France
{FirstName.LastName@inria.fr}

Abstract. Discovering Probabilistic Frequent Itemsets (PFI) in uncertain data is very challenging since algorithms designed for deterministic data are not applicable in this context. The problem is even more difficult for uncertain data streams where massive frequent updates need be taken into account while respecting data stream constraints. In this paper, we propose FMU (Fast Mining of Uncertain data streams), the first solution for exact PFI mining in data streams with sliding windows. FMU allows updating the frequentness probability of an itemset whenever a transaction is added or removed from the observation window. Using these update operations, we are able to extract PFI in sliding windows with very low response times. Furthermore, our method is exact, meaning that we are able to discover the exact probabilistic frequentness distribution function for any monitored itemset, at any time. We implemented FMU and conducted an extensive experimental evaluation over synthetic and real-world data sets; the results illustrate its efficiency.

1. Introduction

Dealing with uncertainty has gained increasing attention these past few years in both static and streaming data management and mining [3, 10, 2, 12, 11]. There are many possible reasons for uncertainty, such as noise occurring when data are collected, noise injected for privacy reasons, semantics of the results of a search engine (often ambiguous), etc. Thus, many sensitive domains now involve massive uncertain data. For instance, scientific applications are producing every day very large sets of experimental and simulation data, so much so that Jim Gray has identified their management and analysis as the "*Fourth Paradigm*" [8]. Example 1 illustrates a collection of uncertain data, where each record is associated to a probability of occurrence.

Example 1. *Recently, China lent two pandas to France (i.e. Huan Huan and Yuan Zi) for ten years. Let us imagine the monitoring of these two pandas by means of sensors. In our scenario, these sensors gather physiological data (blood pressure, temperature, etc.) and transform it into possible activities thanks to a given model. For instance, the rule* $\{pressure = [100..150], temperature = [80..90] \rightarrow sleeping, 75\%\}$ *means that with a blood pressure between 100 and 150mmHg, and a body temperature between 80 and 90 Fahrenheit, the probability that a Panda is sleeping is* 75%. *Figure 1 illustrates the activities*

Huan Huan

e	h	activity	Prob.
1	8	sleeping	0.3
3	9	eating	0.3
5	10	sleeping	0.3
7	11	grooming	0.4
9	12	sleeping	0.3
11	13	drinking	0.3
13	14	courting	0.9
15	15	resting	0.2
17	16	playing	0.4
19	17	growling	0.2

Yuan Zi

e	h	activity	Prob.
2	8	sleeping	0.9
4	9	eating	0.4
6	10	drinking	1
8	11	grooming	0.9
10	12	marking	0.4
12	13	resting	0.2
14	14	climbing	0.2
16	15	courting	0.4
18	16	playing	0.3
20	17	growling	0.9

FIGURE 1. Panda's activities inferred from body sensor data

inferred for the pandas. We can observe, for instance, that Yuan Zi was eating at 9am, with a probability of 40%.

With the probabilistic approach illustrated by Example 1, there are two cases for each uncertain record: either it really occurred in the real world or it did not. A reliable framework for handling such uncertainty lies in the theory of "possible worlds" [6] where each unique combination of records' existence corresponds to a possible world. Unfortunately, there is a combinatorial explosion in the number of possible worlds (n records, each associated to 2 possible values of existence, leading to 2^n possible worlds).

Therefore, in this context, frequent itemset mining [1] must be carefully adapted. Finding the number of occurrences of an itemset X in a database D (also called the support of X in D) is at the core of frequent itemset mining. In the literature, we find two main support measures for uncertain data: Expected Support [5] (an approximate measure of support) and Probabilistic Support [3] (that is an exact measure of support in uncertain data). We work with Probabilistic Support since it gives exact results[1]. And we propose a solution for Probabilistic Frequent Itemset (PFI) mining in data streams using this measure of interest.

There are several ways to observe a data stream, two important ones being batches and sliding windows [9]. Both techniques have pros and cons. Batches allow fast processing but the result is available only after the batch has been fulfilled (which is not compatible with real time constraints). Sliding windows allow maintaining the result any time the stream is updated, but they need more CPU. Today, existing methods for uncertain data stream mining are batch-based and work with Expected Support [19, 12, 11]. However, working with sliding windows is a major matter for numerous monitoring applications where handling "anytime queries" is crucial. Let us consider, for instance, a patient's electrocardiogram that is monitored in real time because alarms must be triggered as soon as an abnormal behavior is detected. This kind of constraints is very important

[1]We give detailed motivations for this choice in Section 6.

when security is at stake (health monitoring, intrusion detection, prediction of natural disasters, etc.) and surveillance must be done in an ongoing fashion. Meanwhile, since a data stream cannot be observed as a whole in main memory, and because related applications call for ongoing processes, an observation window is often needed.

The main challenge, with sliding windows, is to update the support of monitored itemsets upon transaction arrival or removal. We introduce FMU (Fast Mining of Uncertain data streams), a framework adopting the exact approach while meeting the time limitations of data stream environments. *To the best of our knowledge, FMU is the first solution for PFI mining in a sliding window over uncertain data streams.*

Although our approach works on both statistical dependent and independent data (we discuss this point in Section 3) we describe it for independent data for simplicity of presentation. Our contributions are the following:

- We define a new model for uncertain data streams, where an item may have multiple occurrence (each associated to a probability) for one transaction.
- We propose a new approach for computing probabilistic support by recursion on the transactions. This approach allows to develop efficient algorithms for updating probabilistic support after any modification in the sliding window.
- We propose new algorithms for probabilistic frequent itemset mining with sliding windows, where transactions are inserted or deleted. Our algorithms allow updating the new probabilistic support of any monitored itemset with a low complexity since it doesn't need to scan the whole sliding window from scratch.

Our experiments show the feasibility of our approach, which is able to discover and manage PFI in data streams with response time that are up to several orders of magnitude faster than baseline methods of the literature employed in a sliding window context.

2. Problem Definition

We now describe the problem we address with formal definitions of the uncertainty model we adopt, probabilistic itemset mining and uncertain data streams. Our notations are summarized by Figure 5.

2.1. Uncertain Data.

Let I be a set of literals. I is also called the *vocabulary*. An event e_i is a tuple $e_i =< Oid, ts, x, P >$ where i is the identifier of the event, Oid is an object identifier, ts is a timestamp, $x \in I$ is an item and P is an existential probability $P \in [0, 1]$ denoting the probability that e_i occurs.

Example 2. *Consider the data given by Figure 1, the first two events for Huan Huan are:* $e_1 =< Huan\ Huan, 8, sleeping, 0.3 >$ *and* $e_3 =< Huan\ Huan, 9, eating, 0.3 >$.

Definition 1. *An* uncertain item x *is an item that appears in an event, the probability of* x *is the probability of its event.*

Definition 2. *An* uncertain transaction *t is a set of pairs (x, P) for an object such that x is an uncertain item and P is the probability of the event of x. $P(x \in t)$ is the probability of existence of x in t. An* uncertain database *is a set of uncertain transactions.*

Panda	Id	Transaction
Huan Huan	t_1	(eating, 0.3); (sleeping, 0.3)
Yuan Zi	t_2	(eating, 0.4); (drinking, 1)

FIGURE 2. The pandas' activities (uncertain transactions) from 9am to 10am

Example 3. *Figure 2 gives the uncertain transaction database of Huan Huan and Yuan Zi for two hours, from 9am to 10am. We can observe that Yuan Zi's activities in this time window were: eating with a probability of 40% and drinking with a probability of 100%.*

	Possible Worlds	Probability
w_1	{}; {drinking}	0.294
w_2	{eating}; {drinking}	0.126
w_3	{sleeping}; {drinking}	0.126
w_4	{eating, sleeping}; {drinking}	0.054
w_5	{}; {eating, drinking, }	0.196
w_6	{eating}; {eating, drinking, }	0.084
w_7	{sleeping}; {eating, drinking, }	0.084
w_8	{eating, sleeping}; {eating, drinking, }	0.036

FIGURE 3. Possible worlds for the database illustrated in Figure 2

An uncertain database can be treated as a set of deterministic databases, called possible worlds. The possible worlds are generated from the possible instances of transactions. Let w be a possible world, then the instance of a transaction t in w is denoted by t_w. Figure 3 shows the possible worlds for the database in Figure 2. In this database, the instance of transaction t_1 in w_3 is $\{sleeping\}$, and that of transaction t_2 is $\{drinking\}$. For each possible world w, there is probability $P(w)$ that is computed based on the probability of its transaction instances. The sum of the probabilities of all possible worlds of a database is equal to one.

In the case of independence of events, the probability of a given world is computed as $P(w) = \prod_{t \in I} P(t_w)$, where $P(t_w)$ is the probability of t's instance in w. $P(t_w)$ is computed as follows:

$$P(t_w) = (\prod_{x \in t_w} (P(x \in t)) \times (\prod_{x \notin t_w} (1 - P(x \in t)))$$

Intuitively, we multiply the existential probability of t items that are present in t_w by the probability of absence of those that are not present in t_w.

Example 4. *In the possible worlds shown in Figure 3, the probability of w_4 is equal to the occurrence of eating and sleeping for transaction t_1, drinking for t_2, and the non-occurrence of eating for t_2. Thus $P(w_4) = (0.3 \times 0.3 \times 1) \times (1 - 0.4) = 0.054$.*

2.2. **Probabilistic Frequent Itemsets.** The problem of frequent itemset mining from a set of transactions T, as defined in [1], aims at extracting the itemsets that occur in a sufficient number of transactions in T. This is based on the number of transactions in T where an itemset X appears (i.e. the support of X in T). In the deterministic context, computing this support is straightforward (with a scan over T). In uncertain databases, however, the support varies from one possible world to another. For this reason, the support of an itemset in an uncertain database, introduced in [3], is given as a *probability distribution function*. In other words, each possible value $i \in \{0, \ldots, |T|\}$ for the support of X is associated to a probability that is the probability that X has this support in the uncertain database. Definition 3 gives a more formal definition of this notion.

Definition 3. *Let W be the set of possible worlds and $S_{X,w}$ be the support of $X \in I$ in world $w \in W$. The probability $P_{X,T}(i)$ that X has support i in the set of uncertain transactions T is given by:*

$$P_{X,T}(i) = \sum_{w \in W, S_{X,w} = i} P(w) \tag{1}$$

The probability distribution function $P_{X,T}(i)$ for $i \in [0..|T|]$ is called the probabilistic support of X.

Example 5. *In the possible worlds given by Figure 3, we have $P_{eating,T}(1) = P(w_2) + P(w_4) + P(w_5) + P(w_7) = 0.46$. In other words, the probability that exactly one Panda is eating between 9am and 10am is 46%.*

Definition 4. *Given a support value i, the probability $P_{\geq X,T}(i)$ that an itemset X has at least i occurrences in T, is given by:*

$$P_{\geq X,T}(i) = \sum_{j=i}^{|T|} P_{X,T}(j) \tag{2}$$

Given minSup and minProb, a user minimum support and minimum probability, and T a set of uncertain transactions, an itemset X is a probabilistic frequent itemset *(PFI) iff $P_{\geq X,D}(minSup) \geq minProb$. $P_{\geq X,D}(minSup)$ is also called the* frequentness probability *of X.*

Example 6. *Figure 4 gives the probability distribution function of the itemset "eating". The probability that "eating" has support of at least 1 is given by $P_{eating,T}(1) + P_{eating,T}(2) = 0.46 + 0.12 = 0.58$. In other words, the probability that at least one panda was eating between 9am and 10am is 58%.*

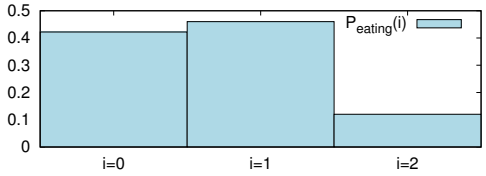

FIGURE 4. The probabilistic support of "eating" in the database of figure 2

Notation	Description
W, w	Set of all possible worlds, and $w \in W$ a possible world
T, t	Set of uncertain transactions, and $t \in T$ a transaction
I	Set of all items
X, x	Itemset $X \subseteq I$, and $x \in X$ an item
$S_{X,w}$	Support (number of occurrences) of X in world w
$P_{X,T}(i)$	Probability that the support of X in the set of transactions T is i
$P_{\geq X,T}(i)$	Probability that the support of X in the set of transactions T is at least i
$P(X \subseteq t)$	Probability that itemset X is a subset of a transaction t

FIGURE 5. Description of Notations

2.3. **Uncertain Data Stream Mining.** In many applications, the data production rate is so high that their analysis in real time with traditional methods is impossible. Sensor networks, Web usage data, scientific instruments or bio-informatics, to name a few, have added to this situation. Because of their rate, data streams should often be observed through a limited observation window and their analysis is highly constrained (*e.g.* "in real-time", "with ongoing queries", "with no access to outdated data", etc.). There are several models for this observation, including sliding windows [18] . Definition 5 gives a formal definition of this notion.

Definition 5. *An* event data stream *(or* data stream*) is an unbounded stream of ordered events. Given n, the maximum number of events to maintain in memory, a* sliding window *over a data stream contains the last n events from the stream.*

The problem of *probabilistic frequent itemset mining in a sliding window* is to extract the set of probabilistic frequent itemsets after each update. The updates occur when a new event is added to the stream and the oldest one is removed from the sliding window.

3. PFI Mining in Sliding Windows

We now introduce FMU, our framework for PFI mining in uncertain data streams with a sliding window SW. FMU allows monitoring the probabilistic support of all the itemsets of SW in real time, as opposed to the batch model where these results are obtained only when a batch is complete. However, the main challenge in this approach consists in updating the probabilistic support of an itemset X when a transaction t is added to, or removed from, the stream. In particular, updating the probabilistic support upon transaction removal is crucial. We give our solution for this step in Section 3.5. In deterministic data, this operation is simple, we just check if $X \subseteq t$ and update its support consequently. In the context of possible worlds, there is no such straightforward approach and the challenge is to update the probabilistic support as fast as possible in order to match the constraints of streaming environments.

Before describing our solution, we mention that one of its requirements is to know $P(X \subseteq t)$, the probability that itemset X is included in transaction t. In the case of independent items, it can be computed as $P(X \subseteq t) = \prod_{i=1}^{|X|} P(x_i \in t)$. In the case where items of transaction t are dependent, for computing $P(X \subseteq t)$ we have to take into account the rules defined on the dependency of items. For example, if two items x_1 and x_2 have a mutual exclusion dependency, then the probability that $X = \{x_1, x_2, \dots\}$ is a subset of a transaction t is zero.

Panda	**Sliding window of size 6, after** e_8
Huan Huan	(eating, 0.3); (sleeping, 0.3); (grooming, 0.4)
Yuan Zi	(eating, 0.4); (drinking, 1); (grooming, 0.9)
	Sliding window of size 6, after e_9
Huan Huan	(eating, 0.3); (sleeping, 0.51); (grooming, 0.4)
Yuan Zi	(eating, 0.4); (drinking, 1); (grooming, 0.9)
	Sliding window of size 6, after e_{10}
Huan Huan	(sleeping, 0.51); (grooming, 0.4)
Yuan Zi	(eating, 0.4); (drinking, 1); (grooming, 0.9); (marking, 0.4)

FIGURE 6. Sliding windows of size 6 from e_3 to e_{10}

3.1. **Sliding Window Model.** Our sliding window model maintains a set of uncertain transactions in memory. When the stream produces a new event $e_i =< Oid, time, x, P >$, the corresponding object in the model is either created or updated in the window. With streaming data, an item x may occur at several points in time and each occurrence is associated to a probability. Therefore, we must give a reliable probability of existence of x, by taking each probability of occurrence into account. To that end, we consider $P(x \in t)$ as the probability that at least one occurrence of x exists in t (*i.e.* 1 minus the probability that x does not exist in t). Let $x_{i_1}, \ldots, x_{i_n}$ be the occurrences of x_i in t, then we compute $P(x \in t)$ as follows : $P(x_i \in t) = 1 - \prod_{j=1}^{n}(1 - P(x_{i_j}, t))$.

Example 7. *Consider the stream of events illustrated in Figure 1 and SW, the sliding window limited to the last 6 events. Figure 6 illustrates the content of SW from 11am (i.e. e_3 to e_8) to 12am (i.e. e_5 to e_{10}). In this example, when e_9 is added, we update the probability of sleeping for Huan Huan but we do not need to remove any item from SW. Then, after e_{10}, we add marking to the uncertain transaction of Yuan Zi and e_3, the oldest event, must be removed.*

3.2. **Computing Frequentness Probability.** For computing the probability that an itemset X is frequent, we need to sum up the probabilities of all supports i for $i > minsup$. In other words, we have $P_{\geq X,T}(minsup) = \sum_{i=minsup}^{|T|} P_{X,T}(i)$, where $P_{X,T}(i)$ is the probability of support i for X in T. Notice that the sum of the probabilities in each row is equal to one.

Therefore, we have:

$$P_{\geq X,T}(minsup) = (1 - \sum_{i=0}^{minsup-1} P_{X,T}(i)) \tag{3}$$

We use Equation 3 for computing the frequentness probability of itemsets. To update the frequentness probabilities after inserting/deleting a transaction, we need to be able to compute and update the probability of support i $(0 \leq i \leq minSup - 1)$ for an itemset X after inserting/deleting a transaction to/from the sliding window.

Our approach for computing the probabilistic support of itemsets uses a recursion on transactions. Using it, we propose our algorithms for updating the probabilistic supports in the sliding window after inserting or deleting transactions.

3.3. **Recursion On Transactions.** Let X be an itemset, DB^n be an uncertain database involving transactions $T = \{t_1, \ldots, t_n\}$, and $P_{X,T}(i)$ be the probability that the support of X, in the set of transactions T, is i. We develop an approach for computing $P_{X,T}(i)$ by doing recursion on the number of transactions.

3.3.1. *Base.* Let us first consider the recursion base. Consider DB^1 be a database that involves only transaction t_1. In this database, the support of X can be zero or one.

The support of X in DB^1 is 1 with probability $P(X \subseteq t_1)$, and its support is 0 with probability $(1 - P(X \subseteq t_1))$. Thus, for the probabilistic support of X in DB^1, we have the following formula:

$$P_{X,\{t_1\}}(i) = \begin{cases} P(X \subseteq t_1) & \text{for i=1;} \\ (1 - P(X \subseteq t_1)) & \text{for i=0;} \\ 0 & \text{for i} > 1 \end{cases}$$

3.3.2. *Recursion Step.* Assume we have DB^{n-1}, a database involving the transactions $t_1, \ldots, t_{n-1}$. We construct DB^n by adding transaction t_n to DB^{n-1}. If $X \not\subseteq t_n$ then the probability of support i for X in DB^n is exactly the same as that in DB^{n-1}. If $X \subseteq t_n$ then two cases can lead to a support of i for X in DB^n:

(1) $X \subseteq t_n$ in DB^n and the support of X in DB^{n-1} is equal to $i - 1$. Thus, we have: $P_{X,T}(i) = P_{X,T-\{t_n\}}(i - 1) \times (P(X \subseteq t_n))$.
(2) $X \not\subseteq t_n$ and the support of X in DB^{n-1} is equal to i. Thus, we have: $P_{X,T}(i) = P_{X,T-\{t_n\}}(i) \times (1 - P(X \subseteq t_n))$.

Then, the probability of support i for X in a database containing $t_1, \ldots, t_n$ is computed based on theorem 1.

Theorem 1. *Given an itemset X and a set of transactions $T = \{t_1, \ldots, t_{n-1}, t_n\}$, the probabilistic support of X in T can be computed based on the probabilistic support in $T - \{t_n\}$ by using the following equation:*

$$\begin{aligned} P_{X,T}(i) = {} & P_{X,T-\{t_n\}}(i - 1) \times (P(X \subseteq t_n)) \\ & + P_{X,T-\{t_n\}}(i) \times (1 - P(X \subseteq t_n)) \end{aligned} \tag{4}$$

Proof. Implied by the above discussion.

3.4. **Updating Probabilistic Support after Inserting a Transaction.** To efficiently support data mining over uncertain data streams, we need to update efficiently the probabilistic support of itemsets after each update. Here, we deal with the insertion of a new transaction to the sliding window. The case of transaction removal will be addressed in Section 3.5.

After inserting a new transaction to the sliding window, the probabilistic support can be updated as follows (see Algorithm 1). Let $P_{X,T}[0..|SW|]$ be an array such that $P_{X,T}[i]$ shows the probability of support i for itemset X in a set of transactions T. $|SW|$ is the maximum support of a transaction in the sliding window, i.e. the size of the window. Given $P_{X,T}$, we generate an array $P_{X,T+\{t\}}$ such that $P_{X,T+\{t\}}[i]$ shows the probability of support i for X in $T + \{t\}$. Algorithm 1 shows the steps for filling the array $P_{X,T+\{t\}}$. It considers two main cases: either T is empty or T is not empty (so $P_{X,T}$ is available). In the first case, we have only one transaction in the sliding window. Thus, our algorithm

initializes $P_{X,T+\{t\}}$ using the base of our recursive formula (described in Section 3.3.1) by setting $P_{X,T+\{t\}}[1] = P(X \subseteq t)$ and $P_{X,T+\{t\}}[0] = 1 - P(X \subseteq t)$. In the second case, i.e. where T is not empty, the algorithm computes the values of $P_{X,T+\{t\}}$ based on those in $P_{X,T}$ by using our recursive formula (i.e. Equation 4) as follows:

$$P_{X,T+\{t\}}[i] = (P_{X,T}[i-1] \times P(X \subseteq t)) + (P_{X,T}[i] \times (1 - P(X \subseteq t)))$$

When $P(X \subseteq t) = 0$ we can simply ignore the transaction since it has no impact on the support, thus we have $P_{X,T+\{t\}} = P_{X,T}$. Recall that for computing the frequentness probability of itemsets, we need to know only the probability of supports between zero and $minSup - 1$. This is the reason why in our algorithm we fill the array only for the values that are lower than *minSup*. Example 8 illustrates our algorithm.

Example 8. *Figure 7 shows the execution of our algorithm over the database shown in Figure 2, with X=eating. Recall that, in this database, we have:* $P(X \subseteq t_1) = 0.3$ *and* $P(X \subseteq t_2) = 0.4$. *Initially* $T = \{\}$, *then we add* t_1 *and afterwards* t_2 *to it. In the fist row, the algorithm sets the probabilistic supports for* $T = \{t_1\}$. *Thus, we have* $P_{X,T+\{t_1\}}[1] = P(X \subseteq t_1) = 0.3$ *and* $P_{X,T+\{t_1\}}[0] = (1 - P(X \subseteq t_1)) = 1 - 0.3 = 0.7$. *The probabilities in the second row are computed using our recursive definition. For example,* $P_{X,\{t_1,t_2\}}[1] = (P_{X,\{t_1\}}[0] \times P(X \subseteq t_2)) + (P_{X,\{t_1\}}[1] \times (1 - P(X \subseteq t_2)) = (0.7 \times 0.4) + (0.3 \times 0.6) = 0.46$.

T				
$\{t_1, t_2\}$	0.42	0.46	0.12	
$\{t_1\}$	0.7	0.3		
	0	1	2	possible supports

FIGURE 7. Computing the probabilistic support of *eating* in the uncertain database of Figure 2

The time complexity of Algorithm 1 for updating the probabilistic support of an itemset X after inserting a new transaction to the sliding window is $O(minsup)$. Its space complexity is $O(|SW|)$ where $|SW|$ is the size of the sliding window, i.e. the maximum number of transactions in the window.

3.5. **Updating Probabilistic Support after Deleting a Transaction.** Assume we have the probabilistic support of an itemset X for a set of transactions T, then the question is: "how to compute the probabilistic support in $T - \{t\}$?" One might think that the probabilistic support i for X in $T - \{t\}$ (*i.e.* $P_{X,T-\{t\}}(i)$) could be computed as $P_{X,T}(i-1)/P(X \subseteq t) + P_{X,T}(i)/(1 - P(X \subseteq t))$. Unfortunately, this formula will not work. For example, if we use it for computing $P_{eating,\{t_1\}}(1)$ after deleting transaction t_2 from the database used in Example 8, then we obtain $0.42 \times 0.4 + 0.46 \times 0.6 = 0.444$,

Algorithm 1 Updating the probabilistic support of an itemset X when a transaction is inserted into the sliding window.

Input: X: itemset; t: new transaction; T: set of transactions before arrival of t; $P_{X,T}$: an array containing probabilistic support of X in T

Output: $P_{X,T+\{t\}}$: an array containing probabilistic supports for X in $T + \{t\}$

1: **if** $|T| = 0$ **then**
2: $P_{X,T+\{t\}}[1] = P(X \subseteq t)$
3: $P_{X,T+\{t\}}[0] = 1 - P(X \subseteq t)$
4: **else**
5: **if** $P(X \subseteq t) = 0$ **then**
6: $P_{X,T+\{t\}} = P_{X,T}$
7: **else**
8: $P_{X,T+\{t\}}[0] = P_{X,T}[0] \times (1 - P(X \subseteq t))$
9: $k = min\{minSup - 1, |T|\}$;
10: **for** $i = 1..k$ **do**
11: $P_{X,T+\{t\}}[i] = (P_{X,T}[i-1] \times P(X \subseteq t)) + (P_{X,T}[i] \times (1 - P(X \subseteq t)))$
12: **end for**
13: **if** $minSup - 1 > |T|$ **then**
14: $P_{X,T+\{t\}}[|T|+1] = (P_{X,T}[|T|] \times P(X \subseteq t))$
15: **end if**
16: **end if**
17: **end if**
18: **return** $P_{X,T+\{t\}}$

whereas the value of $P_{eating,\{t_1\}}(1)$ is equal to 0.3 (see Figure 7). To solve the problem of updating the probabilistic support of X in $T - \{t\}$, we develop the following theorem:

Theorem 2. *Let X be an itemset, T a set of transactions, and $P_{X,T}$ an array denoting the probabilistic support of X in T. Assume we delete a transaction t from T. Let $P_{X,T-\{t\}}(i)$ be the probability for X to have support i in $T - \{t\}$, then $P_{X,T-\{t\}}(i)$ can be computed as:*

$$P_{X,T-\{t\}}(i) = \begin{cases} \frac{P_{X,T}(i) - (P_{X,T-\{t\}}(i-1) \times P(X \subseteq t))}{1 - P(X \subseteq t)} & \text{if } P(X \subseteq t) \neq 1; \\ P_{X,T}(i+1) & \text{otherwise;} \end{cases}$$

Proof. In the case where $P(X \subseteq t) = 1$, it is obvious that by removing t from T, the support of X is reduced by one. Thus, the probability of support i in $T - \{t\}$ is equal to the probability of support $i + 1$ in T. For the case where $P(X \subseteq t) \neq 1$, it is sufficient to show that: $P_{X,T-\{t\}}(i) \times (1 - P(X \subseteq t)) = P_{X,T}(i) - (P_{X,T-\{t\}}(i-1) \times P(X \subseteq t))$.

For this, we expand the right side of this equation by using Equation 4 in Section 3.3.2. We replace $P_{X,T}(i)$ by its equivalent, that is: $P_{X,T-\{t\}}(i-1) \times (P(X \subseteq t)) + P_{X,T-\{t\}}(i) \times (1 - P(X \subseteq t))$

Thus, we have:
$P_{X,T}(i) - (P_{X,T-\{t\}}(i-1) \times P(X \subseteq t))$
$= P_{X,T-\{t\}}(i-1) \times (P(X \subseteq t)) + P_{X,T-\{t\}}(i) \times (1 - P(X \subseteq t)) - (P_{X,T-\{t\}}(i-1) \times P(X \subseteq t))$
$= P_{X,T-\{t\}}(i) \times (1 - P(X \subseteq t))$ □

Theorem 2 suggests to compute $P_{X,T-\{t\}}(i)$ based on $P_{X,T}(i)$ and $P_{X,T-\{t\}}(i-1)$. To develop an algorithm based on this theorem, we need to compute $P_{X,T-\{t\}}(0)$ that is the probability of support 0 for X in $T - \{t\}$. This can be done as follows. We use the fact that when a transaction t is added to the sliding window, the probability of support 0 is multiplied by the probability of absence of t. Thus, when t is removed from T, to compute $P_{X,T-\{t\}}(0)$ we can divide $P_{X,T}(0)$ by (1 - $P(X \subseteq t)$), if $P(X \subseteq t) \neq 1$. In other words, we have:

$$P_{X,T-\{t\}}(0) = \frac{P_{X,T}(0)}{1 - P(X \subseteq t)}, for P(X \subseteq t) \neq 1 \quad (5)$$

Equation 5 works iff $P(X \subseteq t) \neq 1$. In the case where $P(X \subseteq t) = 1$, we can compute $P_{X,T-\{t\}}(0)$ by simply multiplying the probability of absence of all transactions contained in the sliding window. Thus, we have:

$$P_{X,T-\{t\}}(0) = \prod_{t_j \in (T-\{t\})} (1 - P(X \subseteq t_j)) \quad (6)$$

Equation 6 works even in the cases where $P(X \subseteq t) \neq 1$. But, in those cases, we prefer to use Equation 5 because it leads to a more efficient computation of $P_{X,T-\{t\}}(0)$.

Based on Theorem 2 and Equations 5 and 6, we develop Algorithm 2 that updates the probabilistic support after removing a transaction from a sliding window. Recall that for finding frequent itemsets, we need only to compute the probabilistic supports for values that are lower than *minSup*. This is why the "for loop" in the algorithm (started at Line 10) is from 1 to $min\{minSup - 1, |T| - 1\}$. For updating the probabilistic support of an itemset X after deleting a transaction from the sliding window, Algorithm 2 has the same time and space complexity as Algorithm 1.

4. Experiments

We evaluate the performance of FMU by a thorough comparison to existing algorithms in the literature that use Probabilistic Support in exact [3] and approximate [15] mining. Since we do not find sliding window approaches in the literature, we have implemented these algorithms as follows: each time an event is added or removed from the sliding window, the algorithm runs, from scratch, on the content of the updated sliding window. **PFIM** is the algorithm of [3] implemented with all the optimizations (including the 0-1 optimization). However, due to extremely high response time in batch mode, we implemented two other versions of this algorithm. In **PFIM-50%** the discovery is not

Algorithm 2 Updating the probabilistic support of an itemset X after deleting a transaction.

Input: X: itemset; t: deleted transaction; T: set of transactions before delete; $P_{X,T}$: an array containing probabilistic support of X in T

Output: $P_{X,T-\{t\}}$: an array containing probabilistic supports for X in $T - \{t\}$

1: **if** $P(X \subseteq t) = 0$ **then**
2: $P_{X,T-\{t\}} = P_{X,T}$
3: **else**
4: **if** $P(X \subseteq t)$ ¡ 1 **then**
5: $P_{X,T-\{t\}}[0] = \frac{P_{X,T}[0]}{1-P(X \subseteq t)}$
6: **else**
7: $P_{X,T-\{t\}}[0] = \prod_{t_j \in (T-\{t\})} (1 - P(X \subseteq t_j))$
8: **end if**
9: $k = min\{minSup - 1, |T| - 1\}$;
10: **for** $i = 1..k$ **do**
11: **if** $P(X \subseteq t) = 1$ **then**
12: $P_{X,T-\{t\}}[i] = P_{X,T}[i + 1]$
13: **else**
14: $P_{X,T-\{t\}}[i] = \frac{P_{X,T}[i]-(P_{X,T-\{t\}}[i-1] \times P(X \subseteq t))}{1-P(X \subseteq t)}$
15: **end if**
16: **end for**
17: **end if**
18: **return** $P_{X,T-\{t\}}$

performed for each event but for each two events (only 50% of the events are considered). In **PFIM-25%**, the discovery on the sliding window is performed each 4 events. Eventually, **Poisson** is the algorithm of [15] (that allows approximate PFI mining) running on the whole sliding window after each update. A discussion of these algorithms is given in Section 5.

We use two datasets for these experiments: a synthetic one (by the IBM[2] generator) and a real one (the "accident" dataset from the FMI repository[3]). The synthetic dataset contains 38 millions of events, 8 millions of transactions and 100 items. The accidents dataset contains 11 millions of events, 340K transactions and 468 items. We have added an existential probability $P \in]0..1]$ to each event in these datasets, with a uniform distribution. For both datasets, $minSup$ has been set to 30% of the window size and $minProb$ to 40%.

[2] http://www.cs.loyola.edu/~cgiannel/assoc_gen.html
[3] http://fimi.ua.ac.be/data/

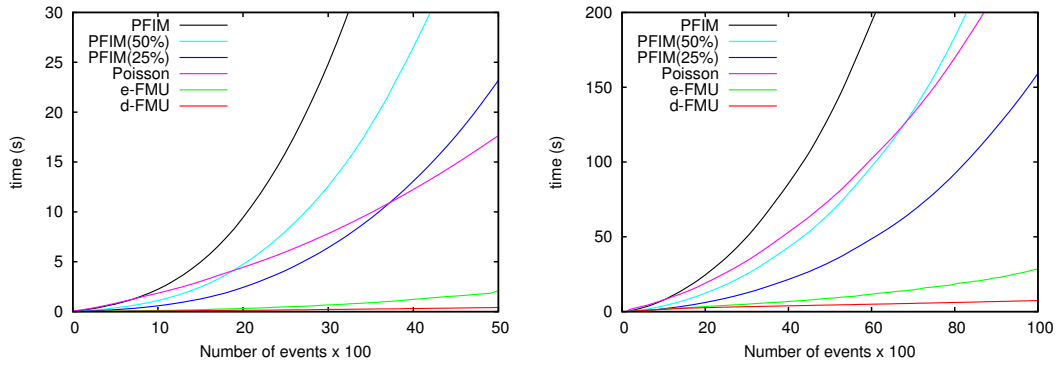

FIGURE 8. Initialization time (filling SW) on synthetic (top) and accident (bottom) datasets

We have implemented two versions of FMU. The first one is "Dynamic-FMU" (**d-FMU** in our experiments). In this version, when a new candidate itemset is generated, it's frequentness probability will be checked over the next updates in the stream thanks to Algorithms 1 and 2. This is the fastest approach but it implies a delay in the pattern discovery (similar to the delay described in [14]). The second version is "Exact-FMU" (**e-FMU** in our experiments). Here, each time a candidate itemset is generated it is immediately verified, from scratch, over all the transactions maintained in the current sliding window with Algorithm 1. Besides that, the probabilistic support of all existing itemsets is maintained at each update thanks to Algorithms 1 and 2. e-FMU guarantees an exact PFI discovery at any point in the stream. However, this is done at the price of a higher time complexity compared to d-FMU.

4.1. **Feasibility.** Figure 8 shows the time needed by each algorithm to extract the PFI in a growing sliding window SW. The size of SW grows from 0 to 5000 transactions for the synthetic dataset and from 0 to 10000 for the accident dataset. This corresponds to the initialization of the stream. We observe that the response time of d-FMU increases barely since it needs very few calculations. e-FMU increases more clearly, since it must scan SW each time a new candidate is proposed. Meanwhile, all the versions of PFIM and Poisson have much higher response times. d-FMU needs 7.34s to fill SW for the accident dataset, where PFIM needs 618s. Furthermore, we can see that Poisson is faster than all versions of PFIM after a number of transactions, but not for the first ones. This is due to the large number of infrequent patterns extracted by Poisson, caused by the approximation of Expected Support. Actually, for the first hundreds of transactions, Poisson may extract up to 146 PFI while the real number of PFI is 36 at most. Such a large number of erroneous PFI is a cause of unnecessary computations and high response times.

Figure 9 shows the time needed by each algorithm to process 100 events, while the transaction data is fed in a pass-through fashion. Although probabilistic supports are maintained

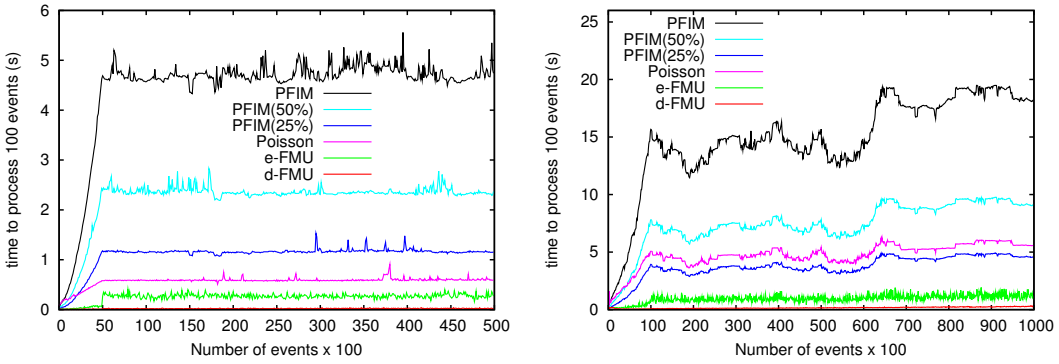

FIGURE 9. Processing times for 100 events on synthetic (top) and accident (bottom) datasets

after each update in the cases of d-FMU, e-FMU, Poisson and PFIM, we report the time for 100 events because the response time of d-FMU, for only one event, would always be 0s. That time is recorded as the number of processed events increases, from the 100^{th} event to the 50000^{th} one in the case of synthetic dataset (100000^{th} for accident dataset). We observe that d-FMU needs less than 0.05s to update the supports of the monitored itemsets in memory for each 100 updates to the stream. e-FMU needs more time (up to 1s) since it has to scan SW when new candidate itemsets are generated. Depending on the dataset, Poisson is faster or slower that PFIM-25%. This is due to the difference in density between these datasets, where Poisson can extract itemsets that are not frequent (slowing down the extraction process). Over the synthetic dataset, the time needed by e-FMU is 5 times faster than Poisson (while extracting exact probabilistic support, whereas Poisson gives an approximation with Expected Support) and up to 20 times faster than PFIM. We also observe that d-FMU is very close to 0s. In fact, in our experimental data, d-FMU appears to run up to two orders of magnitude faster than PFIM on the accident dataset to process 100 events. The global response time of d-FMU, as the stream passes through, is several orders of magnitude lower than that of PFIM.

4.2. **Scalability.** Figure 10 shows the running times of each algorithm for a full sliding window. More precisely, when a sliding window SW is full (after initialization), we measure the time needed to process $|SW|$ events. This time is measured for an increasing size of SW. Our experiments clearly show that d-FMU incurs very few overhead to the computations needed for maintaining the data structures. We can observe, for instance, that when $|WS| = 5000$ in the synthetic dataset, e-FMU needs 14s to process 5000 events, when Poisson takes 29s. These results confirm those shown in Figure 9 where Poisson needs approximately 0.6s (and approximately 0.3s for e-FMU) to process 100 events.

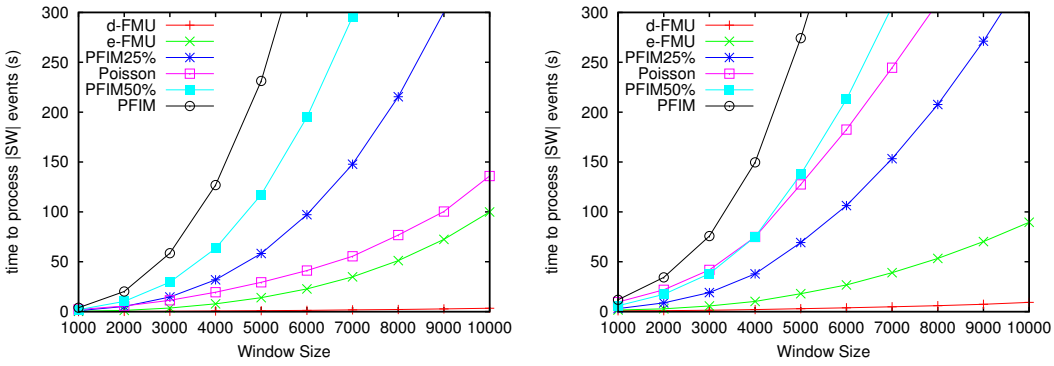

FIGURE 10. Processing time for $|SW|$ events with increasing size of SW on synthetic (top) and accident (bottom) datasets

5. Related Work

Expected Support. Uncertain data mining is a recent research topic that is gaining increasing attention [10, 4, 13, 11, 17]. In [5], the problem of itemset mining from uncertain data is introduced and the authors propose the notion of *Expected Support* as a first solution. Let $P(X \subseteq t)$ be the probability that itemset X is included in transaction t, the Expected Support $ES(X)$ of X in database D is given by: $ES(X) = \sum_{j=1}^{|D|} P(X, t_j)$. This support is then used as a frequency measure (compared to a user minimum threshold) in U-Apriori, a level-wise approach based on the Apriori principle for frequent itemset mining.

Probabilistic Support. In [3], the authors introduce the notion of *probabilistic support* which is an exact measure of an itemset support in the possible world model. Their idea is to find the probability that an itemset X has support i by using Definition 4. The authors propose to compute the frequentness probability of an itemset X with a dynamic programming approach inspired from [16]. Actually, a number of contributions to probabilistic data management and querying problems have shown the relevance of dynamic programming for this purpose [16, 2, 3]. These papers rely on a divide and conquer approach that avoids enumerating all the possible worlds. The principle is to consider processing a request on a dataset T as a recursion on subsets of T. This principle has been applied to the problems of Top_k queries [16] or aggregate queries [2] in uncertain databases. It has also been exploited in [3] in order to avoid enumerating the whole set of possible worlds and speed-up the extraction of exact probabilistic frequent itemsets. However, their approach is incremental in the support (*i.e.* the transaction set is fixed and each iteration of their recursion allows computing the support probability of an itemset for an increasing support). Therefore, we believe that adapting this method to data streams would be very difficult, since *we need a method that would be incremental in*

the transactions as proposed in Section 3.4 (each iteration should allow computing the support probability after reading a new transaction).

Some approximation methods for the probabilistic support of an itemset have also been proposed. The idea of [15] is to approximate the support distribution function by means of a Poisson law. In [4], the authors propose another approximation of frequentness probability based on the central limit theorem. The main drawbacks of these approaches are to use Expected Support as a measure of probabilistic frequentness [15] and to work only on statistical dependent data [15, 4]. We discuss these points in Section 6.

Uncertain Data Streams. Itemset mining in data streams is an important topic of knowledge discovery [14, 7]. Mainly, we find contributions on the extraction techniques and the data models, such as batches [7] or sliding windows [14, 18]. In [9], we find a comparative study of these models. In [19], the authors propose to extract frequent items in probabilistic data. Their approaches allow finding items (itemsets of only one item) in static data and likely frequent items in data streams. [12] proposes to extract frequent itemset from streaming uncertain data by means of Expected Support and a batch model. In [11], we find a batch-based approach to extract frequent itemsets using Expected Support in uncertain data streams with a technique inspired from [7].

Despite the interest of exact PFI mining with sliding windows [9, 18], we do not find any proposal in the literature for such an approach. As we discuss in Section 3, the main challenge in this context is to update the probabilistic support of an itemset when a transaction is added to or removed from the window. Our work is therefore motivated by the needs and challenges of providing an approach that is able to i) extract PFI from data streams; ii) use sliding windows and update the support of an itemset upon transaction insertion or removal; and iii) work with statistical dependent and independent data.

6. Limitations of Expected Support and Approximate Approaches

In the literature, we find interesting approaches using Expected support or estimating the Probabilistic Support of an itemset. These approaches have low complexity, allowing fast processing, but building on such estimations has drawbacks that motivate us for proposing a method that works with Probabilistic Support.

Let us consider the support count of *sleeping* in the Pandas' activities given by Figure 1. Sleeping is very important for Pandas and monitoring the frequency of this activity allows preventing health problems. The Expected Support of *sleeping* for Huan Huan is $0.3 + 0.3 + 0.3 = 0.9$ and it is the same for Yuan Zi (only one occurrence, with a probability of 0.9). Therefore, with the same parameters, if *sleeping* is found frequent for Huan Huan under Expected Support, then it will also be found frequent for Yuan Zi. The fact that Huan Huan is sleeping three times a day and Yuan Zi only once does not make any

difference for Expected Support. Furthermore, by using the same parameters that make *sleeping* a frequent item for both pandas' activities, we will find that *drinking*, *grooming* and *growling* are also frequent for Yuan Zi (as well as *sleeping* and *courting* for Huan Huan).

Another issue of approximate approaches [15, 4] is that they can not work correctly on dependent data. Actually, many real world applications involve dependent data and require careful attention. Consider road traffic monitoring applications and speed cameras. It is possible that an observed vehicle is a tractor. It is also possible that a vehicle speed is 90Mph. However, it is not possible that a tractor is observed at 90Mph. Therefore, a world w containing a transaction t where a vehicle is a tractor and the speed is 90Mph does not exist. This probability of zero can be given to the model of exact approaches, whereas existing approximate methods cannot take it into account.

7. Conclusion

In this paper, we proposed FMU, the first solution for exact PFI mining in data streams with sliding windows. FMU allows computing the exact probabilistic support of an itemset whenever a transaction is added or removed from the observation window. Compared to non-incremental algorithms, that need to scan the whole sliding window after each update, our approach shows very low time complexity. Through an extensive experimental evaluation on synthetic and real datasets, we observed that FMU is up to several orders of magnitude faster than a traditional approach, adapted to sliding windows,

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216, June 1993.

[2] R. Akbarinia, P. Valduriez, and G. Verger. Efficient Evaluation of SUM Queries Over Probabilistic Data. *IEEE Transactions on Knowledge and Data Engineering*, To appear, 2012, (http://hal-lirmm.ccsd.cnrs.fr/lirmm-00652293).

[3] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zuefle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 119–128, New York, NY, USA, 2009. ACM.

[4] T. Calders, C. Garboni, and B. Goethals. Approximation of frequentness probability of itemsets in uncertain data. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 749–754, Washington, DC, USA, 2010. IEEE Computer Society.

[5] C.-K. Chui, B. Kao, and E. Hung. Mining frequent itemsets from uncertain data. In *Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'07, pages 47–58, Berlin, Heidelberg, 2007. Springer-Verlag.

[6] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16:523–544, October 2007.

[7] C. Giannella, J. Han, J. Pei, X. Yan, and P. Yu. *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. AAAI/MIT, 2003.

[8] A. J G Hey, S. Tansley, and K. M. Tolle, editors. *The fourth paradigm : data-intensive scientific discovery*. Redmond, Wash. : Microsoft Research, 2009.

[9] P. Kranen and T. Seidl. Harnessing the strengths of anytime algorithms for constant data streams. *Data Min. Knowl. Discov.*, 19:245–260, October 2009.

[10] C. K.-S. Leung and D. A. Brajczuk. Efficient algorithms for the mining of constrained frequent patterns from uncertain data. *SIGKDD Explor. Newsl.*, 11:123–130, May 2010.

[11] C. K.-S. Leung and F. Jiang. Frequent itemset mining of uncertain data streams using the damped window model. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 950–955, New York, NY, USA, 2011. ACM.

[12] C.-S. Leung and B. Hao. Mining of frequent itemsets from streams of uncertain data. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 1663 –1670, 29 2009-april 2 2009.

[13] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng. Mining uncertain data with probabilistic guarantees. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 273–282, New York, NY, USA, 2010. ACM.

[14] W.-G. Teng, M.-S. Chen, and P. S. Yu. A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, pages 93–104, 2003.

[15] L. Wang, R. Cheng, S. D. Lee, and D. Cheung. Accelerating probabilistic frequent itemset mining: a model-based approach. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 429–438, New York, NY, USA, 2010. ACM.

[16] K. Yi, F. Li, G. Kollios, and D. Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. on Knowl. and Data Eng.*, 20:1669–1682, December 2008.

[17] Ying-Ho and Liu. Mining frequent patterns from univariate uncertain data. *Data and Knowledge Engineering*, 71(1):47 – 68, 2012.

[18] C. Zhang, F. Masseglia, and Y. Lechevallier. ABS: The anti bouncing model for usage data streams. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1169–1174, Washington, DC, USA, 2010. IEEE Computer Society.

[19] Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 819–832, New York, NY, USA, 2008. ACM.