Rare Pattern Mining on Data Streams

David Huang, Yun Sing Koh, and Gillian Dobbie

Department of Computer Science, University of Auckland
dhua035@aucklanduni.ac.nz,
{ykoh,gill}@cs.auckland.ac.nz

Abstract. There has been some research in the area of rare pattern mining where the researchers try to capture patterns involving events that are unusual in a dataset. These patterns are considered more useful than frequent patterns in some domain, including detection of computer attacks, or fraudulent credit transactions. Until now, most of the research in this area concentrates only on finding rare rules in a static dataset. There is a proliferation of applications which generate data streams, such as network logs and banking transactions. Applying techniques for static datasets is not practical for data streams. In this paper we propose a novel approach called Streaming Rare Pattern Tree (SRP-Tree), which finds rare rules in a data stream environment using a sliding window, and show that it is faster than current approaches.

Keywords: Rare Pattern Mining, FP-Growth, Data Streams, Sliding Window.

1 Introduction

Traditionally pattern mining techniques focus on finding frequent patterns within a dataset. However in some scenarios rare patterns may be more interesting as they represent unexpected phenomenons. Rare patterns are patterns that do not occur frequently within the dataset and can be considered as exceptions. An example of a useful rare pattern in real life could be the association of certain occurences of symptoms to diseases. For instance, Meningitis is the inflammation of the protective membranes covering the brain and spinal cord. Symptoms of Meningitis include headache, fever, vomiting, neck stiffness, and altered consciousness. Most of the symptoms are commonly occuring for the influenza except for neck stiffness. By discovering the rare occurrence of neck stiffness, we are capable of flagging a patient possibly suffering from Meningitis out of a pool of patients suffering from the common influenza. In the recent years, the problem of extracting rare patterns from static datasets has been addressed. However as the capability to generate data streams increases, the ability to capture useful information from these data is necessary.

Ever since its inception, data stream mining has remained one of the more challenging problems within the data mining discipline. Data from a wide variety of application areas ranging from online retail applications such as online auctions and online bookstores, telecommunications call data, credit card transactions, sensor data and climate data are a few examples of applications that generate vast quantities of data on a continuous basis. Data produced by such applications are highly volatile with new patterns

A. Cuzzocrea and U. Dayal (Eds.): DaWaK 2012, LNCS 7448, pp. 303–314, 2012.

and trends emerging on a continuous basis. The unbounded size of data streams is considered the main obstacle to processing data streams. As it is unbounded, it makes it infeasible to store the entire data on disk. Furthermore, we would want to process data streams near real time. This raises two issues. Firstly, a multi-pass algorithm, entire dataset needs to be stored before mining can commence. Secondly, obtaining the extact set of rules that includes both frequent and rare rules from the data streams is too expensive.

To capture these types of rules, we propose a novel technique called Streaming Rare Pattern Tree (SRP-Tree), which requires a single pass through the dataset using a sliding window approach. We also propose a novel data structure called the Connection Table which allows us to efficiently constrain the search in our tree, and keep track of items within the window. The paper is organized as follows. In Section 2 we look at related work in the area of rare association rule mining. In Section 3 we present preliminary concepts and definitions for rare pattern mining in data streams. In Section 4 we describe our SRP-Tree approach, and in Section 5 we describe and discuss our experimental results. Finally, Section 6 concludes the paper.

2 Related Work

There has been a lot of work in the area of rare pattern mining. However all current research in this area is designed for static datasets and not able to handle a data stream environment. Currently there are two different types of rare pattern mining approaches: level-wise and tree based. Current rare itemset mining approaches which are based on level-wise exploration of the search space are similar to the Apriori algorithm [2]. In Apriori, k-itemsets (itemsets of cardinality k) are used to generate $k + 1$-itemsets. These new $k+1$-itemsets are pruned using the downward closure property, which states that the superset of a non-frequent itemset cannot be frequent. Apriori terminates when there are no new $k + 1$-itemsets remaining after pruning. MS-Apriori [12], Rarity [16], ARIMA [14], AfRIM [1] and Apriori-Inverse [8] are five algorithms that detect rare itemsets. They all use level-wise exploration similar to Apriori, which have candidate generation and pruning steps.

MS-Apriori [12] uses a bottom-up approach similar to Apriori. In MS-Apriori, each item can be assigned a different minimum item support value (MIS). Rare items can be assigned a low MIS, so that during candidate pruning, itemsets that include rare items are more likely to be retained and participate in rule generation. Apriori-Inverse [8] is used to mine perfectly rare itemsets, which are itemsets that only consist of items below a maximum support threshold (maxSup).

Szathmary et al. [14] proposed two algorithms that can be used together to mine rare itemsets: MRG-Exp and ARIMA. They defined three types of itemsets: minimal generators (MG), which are itemsets with a lower support than its subsets; minimal rare generators (MRG), which are itemsets with non-zero support and whose subsets are all frequent; and minimal zero generators (MZG), which are itemsets with zero support and whose subsets all have non-zero support. The first algorithm, MRG-Exp, finds all MRG by using MGs for candidate generation in each layer in a bottom up fashion. The MRGs represent a border that separates the frequent and rare itemsets in the search space.

All itemsets above this border must be rare according to the antimonotonic property. The second algorithm, ARIMA, uses these MRGs to generate the complete set of rare itemsets. ARIMA stops the search for non-zero rare itemsets when the MZG border is reached, which represents the border above which there are only zero rare itemsets.

Adda et al. [1] proposed AfRIM which begins with the itemset that contains all items found in the database. Candidate generation occurs by finding common k-itemset subsets between all combinations of rare $k+1$-itemset pairs in the previous level. Troiano et al. proposed the Rarity algorithm that begins by identifying the longest transaction within the database and uses it to perform a top-down search for rare itemsets, thereby avoiding the lower layers that contain only frequent itemsets.

All of the above algorithms use the fundamental generate-and-test approach used in Apriori, which has potentially expensive candidate generation and pruning steps. In addition, these algorithms attempt to identify all possible rare itemsets, and as a result require a significant amount of execution time. RP-Tree algorithm was proposed by Tsang et al. [17] as a solution to these issues. RP-Tree avoids the expensive itemset generation and pruning steps by using a tree data structure, based on FP-Tree [7], to find rare patterns. RP-Tree finds rare itemsets using a tree structure. However it uses a multi-pass approach, which is not suitable in a data stream environment.

There has been much work in the area of data stream mining, but most work focuses on capturing frequent patterns [4,5,9,3,10,11,13,15]. Up until now there has been no research into rare pattern mining in data streams.

3 Preliminaries

In this section, we provide definitions of key terms that explain the concepts of frequent pattern mining in a data stream. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of literals, called items, that represent a unit of information in an application domain. An set $X = \{i_l, \ldots, i_m\} \subseteq I$ and $l, m \in [1, n]$, is called a itemset, or a k-itemset if it contains k items. A transaction $t = (tid, Y)$ is a tuple where tid is a transaction-id and Y is a pattern. If $X \subseteq Y$, it is said that t contains X or X occurs in t. Let size(t) be the size of t, i.e., the number of items in Y.

A data stream $\mathcal{DS}$ can formally be defined as an infinite sequence of transactions, $\mathcal{DS} = [t_1, t_2, \ldots, t_m)$, where $t_i, i \in [1, m]$ is the ith transaction in data stream. A window $\mathcal{W}$ is a set of all transactions between the ith and jth (where $j > i$) transactions and the size of $\mathcal{W}$ is $|W| = j - i$. The count of a itemset X in a $\mathcal{W}$, denoted as $count_{\mathcal{W}}(X)$, is the number of transactions in $\mathcal{W}$ that contain X, and the support of an itemset X, denoted as $sup_{\mathcal{W}}(X) = \frac{count_{\mathcal{W}}(X)}{|\mathcal{W}|}$.

An association rule is an implication $X \rightarrow Y$ such that $X \cup Y \subseteq \mathcal{I}$ and $X \cap Y = \emptyset$. X is the antecedent and Y is the consequent of the rule. The *support* of $X \rightarrow Y$ in $\mathcal{W}$ is the proportion of transactions in $\mathcal{W}$ that contains $X \cup Y$. The *confidence* of $X \rightarrow Y$ is the proportion of transactions in $\mathcal{W}$ containing X that also contains Y.

3.1 Rare Itemsets

We adopted the rare itemsets concept from Tsang et al. We consider an itemset to be rare when its support falls below a threshold, called the minimum frequent support

(minFreqSup) threshold. One difficulty when generating rare itemsets is differentiating noisy itemsets from the actual rare itemsets. As the support of the itemset is low, the potential of pushing unrelated items together increases as well, thus producing noisy itemsets. To overcome this problem, we define a noise filter threshold to prune out the noise called the minimum rare support (minRareSup) threshold. Typically minRareSup is set to a very low level, e.g., 0.01%.

Definition 1. *An itemset x is a* rare itemset *in a window $\mathcal{W}$ iff*

$$sup_{\mathcal{W}}(x) \leq \textit{minFreqSup and } sup_{\mathcal{W}}(x) > \textit{minRareSup}$$

However not all rare itemsets that fulfill these properties are interesting. Furthermore, rare itemsets can be divided into two types: *rare item itemsets* and *non-rare item itemsets*.

Rare item itemsets refer to itemsets which are a combination of only rare items and itemsets that consist of both rare and frequent items. Given 4 items $\{a, b, c, x\}$ with supports a = 0.70, b = 0.45, c = 0.50, and x = 0.10, with minFreqSup = 0.15 and minRareSup = 0.01, the itemset $\{a, x\}$ would be a rare item itemset assuming that the support of $\{a, x\} > 0.01$, since the itemset includes the rare item x.

Definition 2. *An itemset x is a* rare item itemset *iff*

$$\exists x \in X, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}$$

Non-rare item itemsets only has frequent items which fall below the minimum frequent support threshold. Given 4 items $\{a, b, c, x\}$ with supports a = 0.70, b = 0.45, c = 0.50, and x = 0.10, with minFreqSup = 0.15 and minRareSup = 0.01, and the itemset $\{a, b, c\}$ had a support of 0.09, then this itemset would be a non-rare item itemset as all items within the itemset are frequent, and its support lies between minFreqSup and minRareSup.

Definition 3. *An itemset X is a* non-rare item itemset *iff*

$$\forall x \in X, sup_{\mathcal{W}}(x) > \textit{minFreqSup}, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}$$

4 SRP-Tree: Rare Pattern Tree Mining for Data Streams

In this section we will be discussing the details of our proposed technique SRP-Tree that mines rare patterns in a data stream using a sliding window approach.

4.1 SRP-Tree

Current tree based rare pattern mining approaches follow the traditional FP-Tree [7] approach. It is a two-pass approach and is affordable when mining a static dataset. However in a data stream environment, a two-pass approach is not suitable. To process a static environment, traditional non-streaming rare pattern techniques, such as RP-Tree order the items within a transaction according to its frequencies before inserting it into

the tree. Using these algorithms the frequency of the items are obtained during the first pass through the dataset.

In data streams we can only look at the transactions within the stream once, thus, a one-pass approach is necessary. This rules out the possibility of building a tree based on the frequency of items within the data stream. Furthermore, frequency of an item may change as the stream progresses. There are four scenarios which we need to consider in a stream:

Scenario 1. A frequent item x at particular time T_1 may become rare at time T_2.
Scenario 2. A rare item x at particular time T_1 may become frequent at time T_2.
Scenario 3. A frequent item x at particular time T_1 may remain frequent at time T_2.
Scenario 4. A rare item x at particular time T_1 may remain rare at time T_2.

T_1 represents a point in time, and T_2 represents a future point in time after T_1.

To find rare patterns within data streams, we propose a new algorithm called SRP-Tree. Our algorithm uses a tree based approach. In our approach, items from incoming transactions in the stream are inserted into a tree based on a canonical ordering. A canonical ordering allows us to capture the content of the transactions from the data stream and orders tree nodes according to a particular order. In our approach we use appearance order of the items as the canonical ordering. When we use a canonical order to build the tree, the ordering of items is unaffected by the changes in frequency caused by incremental updates. There has been work carried out using a canonical ordering to build trees for a data stream mining environment [6,10]. But all of these research have been tailored to find frequent patterns.

In our approach, the frequency of a node in the tree is at least as high as the sum of frequencies of its children. However, this does not actually guarantee the overall downward closure property which exists in a frequency ordered tree. The downward closure property in a traditional rare pattern tree mining algorithm, whereby, *rare-items will never be the ancestor of a non-rare item in the initial tree due to the tree construction process* is violated. Hence, we propose a novel item list called the Connection Table which keeps track of each unique item in the window and the items they co-occur with along with their respective frequencies.

4.2 Connection Table

The Connection Table used in our SRP-Tree captures in the transactions only items that have a lower canonical ordering. For example, given a transaction in a canonical order of $\{a, b\}$ we store in the table that a is connected to b with a frequency of 1 but it does not store b is connected to a. This is because of the properties of the canonical ordering in the constructed tree: item a will always be the ancestor of item b. Since mining is carried out using a bottom-up approach, by mining item b, item a is also mined. The opposite does not hold since mining item a does not guarantee that item b is mined. Therefore, by using the Connection Table to keep track of connected items and adding them as arguments to FP-Growth in the mining phase, the complete set of rare-item itemsets can then be captured by SRP-Tree. The Connection table is designed using a hash map which allows for $O(1)$ access. In worst case scenarios, the table could reach

a max size of $\frac{x(x+1)}{2}$ with x being the total amount of items; however, in reality this is highly unlikely.

At any point of time should we decide to mine the current window, SRP-Tree uses the initial tree of the current window to construct conditional pattern bases and conditional trees for each rare-item and their connected items in the Connection Table. Note that only connection items with an occurring frequency greater than or equal to the minRareSup is included. Each conditional tree and corresponding item are then used as arguments for FP-Growth. The threshold used to prune items from the conditional trees is minRareSup. The union of the results from each of these calls to FP-Growth is a set of itemsets that contains a rare-item, or rare item itemsets.

Example. Given the dataset in Table 1, we show how the Connection Table is built in Table 2. The left column in Table 2 list the unique items in the window, whereas the right column list the set of co-occurring items along with the co-occurrence frequency of that particular item to the item in the right column. For example, item c co-occurs twice with items d, f, and h.

Table 1. Set of transactions in a given window $\mathcal{W}$ of size 12

Tid	Transaction	Tid	Transaction	Tid	Transaction
1	{a, g, h}	5	{c, f, h}	9	{c, d, h}
2	{a, g, h, i}	6	{a, g, h, e}	10	{b, f}
3	{b, c, d, f}	7	{g}	11	{a, h}
4	{b, d, j}	8	{h}	12	{a}

Table 2. Connection Table using the window of transactions listed in Table 1

Item	Items Co-occurred	Item	Items Co-occurred
a	{(e:1), (g:1), (h:1), (i:1)}	f	{(h:1)}
b	{(c:1), (d:2), (f:2), (j:1)}	g	{(h:1), (i:1)}
c	{(d:2), (f:2), (h:2)}	h	{(i:1)}
d	{(f:1), (h:1), (j:1)}	i	{∅}
e	{(g:1), (h:1)}	j	{∅}

4.3 SRP-Tree Algorithm

Our SRP-Tree algorithm is shown in Algorithm 1. SRP-Tree essentially performs in one pass the counting of item frequencies and the building of the initial tree. Therefore, in a given window $\mathcal{W}$, for each incoming transaction t, SRP-Tree first updates the list of item frequencies according to the transactions contained in the current window. We refer to this as updateItemFreqList(t) method, where we increment the counts of items contained in the new transaction and decrement the counts of items contained in the oldest transcations to be discarded from the window. SRP-Tree then updates the tree structure according to the transaction contained in the current window in a similar fashion, which is referred to as updateTree(t) method in Algorithm 1.

We mine the tree after a particular number of transactions also known as a block. We refer to a preset block size as $\mathcal{B}$. In this algorithm, $\mathcal{R}$ refers to the set of rare items and $\mathcal{C}$ refers to the set of items that co-occur with a particular rare item.

We would also like to point out, that another difference between SRP-Tree and a static rare pattern mining approach is that in SRP-Tree, the tree is built using all the transactions in the window, whereas in a static rare pattern mining approach, only transactions with rare items are used to build the tree. A static approach has the luxury of looking at the dataset twice and discarding items which it is not interested in before the tree is even built. In our case, we simply cannot know which transactions contain rare items until we decide to mine.

Algorithm 1. SRP-Tree

Input: $DS, \mathcal{W}, \mathcal{B}, minRareSup, minFreqSup$;
Output: $results$ (Set of rare item itemsets);

while exist(DS) **do**
  $t \leftarrow$new incoming transaction from DS;
  currentBlockSize $\leftarrow$ currentBlockSize + 1;
  updateItemFreqList(t);
  updateConnectionTable(t);
  $tree \leftarrow$ updateTree(t);
  Mining at the end of block
  if $\mathcal{B}$ == currentBlockSize **then**
    currentBlockSize $\leftarrow$ 0;
    $results = \emptyset$;
    $\mathcal{I} \leftarrow$ {all unique items in $\mathcal{W}$};
    $\mathcal{R} \leftarrow \{i \in \mathcal{I} \mid sup_{\mathcal{W}}(i) \geq minRareSup \wedge sup_{\mathcal{W}}(i) < minFreqSup\}$;
    $\mathcal{C} \leftarrow \{k \in \mathcal{R}, j \in connectionTable(k) \mid sup_{\mathcal{W}}(k) \geq minRareSup\}$;
    for item a in $tree$ **do**
      if $a \in \mathcal{R}$ or $a \in \mathcal{C}$ **then**
        construct a's conditional pattern-base and then a's conditional FP-Tree $Tree_a$;
        $results \leftarrow results \cup$ FP-Growth$(Tree_a, a)$;
      end if
    end for
  end if
end while
return $results$;

SRP-Tree Example. Applying SRP-Tree to the window $\mathcal{W}$ of the 12 transactions in Table 1, the support ordered list of all items is $\langle(h{:}6), (a{:}5), (g{:}4), (c{:}3), (d{:}3), (b{:}2), (f{:}2), (e{:}1), (i{:}1), (j{:}1)\rangle$. Using minFreqSup = 4 and minRareSup = 2, only the items $\{b, c, d, f\}$ are rare, and included in the set of rare items, $\mathcal{R}$.

The initial SRP-Tree constructed for window W of size 12 is shown in Figure 1. To find the rare-item itemsets, the initial SRP-Tree is used to build conditional pattern bases and conditional SRP-Trees for each rare item $\{b, c, d, f\}$ and any additional items in the Connection Table that is connected with a rare item that has a frequency greater

than the minRareSup, in this example, item h. The conditional tree for item h is shown in Figure 2. Each of the conditional SRP-Trees and the conditional item are then used as parameters for the FP-Growth algorithm.

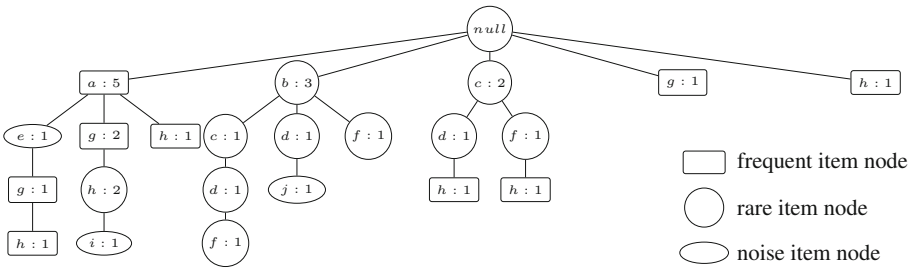

Fig. 1. Pattern tree constructed from window $\mathcal{W}$ using SRP-Tree

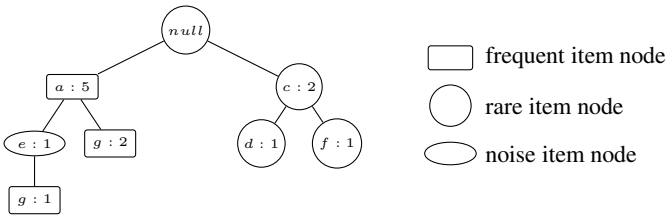

Fig. 2. Conditional tree, $Tree_h$

5 Experimental Results

In our experiments we compared the performance of our SRP-Tree to the Streaming Canonical Tree (SC-Tree), which is a modified version of DSTree [10] with pruning of frequent itemsets. SRP-Tree is the very first attempt at mining rare patterns in a data stream environment and there are no other techniques that mine rare patterns in a data stream to compare to, so we have compared our SRP-Tree to the technique that we call the SC-Tree with pruning which finds rare item itemsets in an unoptimized brute-force manner. The SC-Tree is a one pass technique which stores the transactions from the stream in a tree using the canonical ordering and stores/updates item frequencies like that of our SRP-Tree. One of the major differences of SC-Tree is that SC-Tree does not store or keep track of items in the Connection Table. To find all rare item itemsets, SC-Tree finds and generates all itemsets that meet the minRareSup threshold, then removes all other itemsets except rare item itemsets with an extra pruning step. SC-Tree produces the same itemsets and generates the same rules as SRP-Tree. All algorithms were implemented in Java and executed on a machine equipped with an Intel Core i5-2400 CPU @ 3.10 GHz with 4GB of RAM running Windows 7 x 64.

5.1 Real-World Dataset

In this section we present the time and relative time taken for itemset generation of SRP-Tree and SC-Tree. The time is reported in seconds and the relative time is calculated by setting RP-Tree to 1.00 and SCTree relative to that of RP-Tree.

$$\text{relative time} = \frac{\text{Time taken by SC-Tree}}{\text{Time taken by SRP-Tree}}$$

We used a window size and block size of 25K for all datasets except for the Mushroom dataset where we used 2K due to its smaller size. The minFreqSup and minRareSup thresholds are shown in Table 3 for each dataset. The thresholds are user-defined through examining the distribution of item frequencies in each of the datasets. We acknowledge that given a data stream environment, this is not the most suitable way of defining thresholds and in the future we will be looking at a way to adapt the thresholds to the change in distribution and drift of the transactions in the stream.

We have tested the algorithms on 6 datasets from the FIMI (Frequent Itemset Mining Implementations) repository: Mushroom, Retail, BMS-POS, T10I4D100K, T40I10D100K, and Kosarak (250K).

Table 3. Comparision between SRP-Tree and SC-Tree

Dataset	$\mathcal{B}$	MinRareSup	MinFreqSup	No. of Itemset	SRP-Tree		SC-Tree	
					Time (s)	Rel. Time	Time (s)	Rel. Time
Mushroom	2K	0.01	0.05	14443674	1131	1.00	1321	1.17
Retail	25K	0.0001	0.0005	572673	239	1.00	339	1.42
BMS-POS	25K	0.0002	0.0005	1426	58	1.00	3783	65.22
T10I4D100K	25K	0.0001	0.0005	1161	12	1.00	19	1.58
T40I10D100K	25K	0.003	0.05	4734806	301	1.00	380	1.26
Kosarak (250K)	25K	0.001	0.15	35623519	703	1.00	7213	10.26

Table 3 shows, for each dataset, the block size, minRareSup, and minFreqSup used to run the algorithms then the comparison between SRP-Tree and SC-Tree of the itemsets generated, time it took to run the algorithm, and the relative time. Both SRP-Tree and SC-Tree generate the same amount of itemsets because SRP-Tree only generates rare-item itemsets and SC-Tree generates all itemsets that meet the minRareSup threshold then the extra pruning step removes all other itemsets that are not rare-item itemsets. Therefore, the final number of itemsets generated by both algorithms is the same.

In all datasets of varying item frequency distribution, SRP-Tree runs faster than SC-Tree. The time taken to run the various datasets are highly dependent on the tree structure built from the transactions in the datasets and generally have a positive correlation with the number of itemsets generated. The number of itemsets generated also has a great variation and is highly dependent on the composition/nature of the respective dataset. For datasets that are more sparse in nature like Retail, BMS-POS, and T10I4D100K, the number of itemsets generated are usually than a dense dataset like Mushroom and the run-time is usually faster.

5.2 Case Study: T40I10D100K

The T40I10D100K dataset is generated using the generator from the IBM Almaden Quest research group. Figure 3 and Figure 4 shows the item frequency distribution of the T40I10D100K dataset. When we compared the distribution of the T40I10D100K dataset, to other datasets in the FIMI repository, we observed that the T40I10D100K dataset is more sparse in nature compared to the Mushroom dataset, but more dense than datasets like Retail and BMS-POS. It is also important to note that T40I10D100K contains a high proportion of items with a frequency of less than 0.1 (approximately 90% of the total items). This particular distribution contains a greater number of prospective rare items and rules to be mined from the dataset.

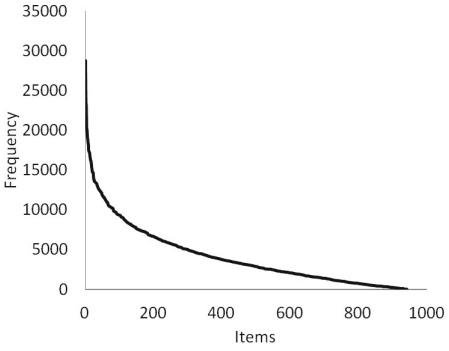

Fig. 3. Item frequency distribution

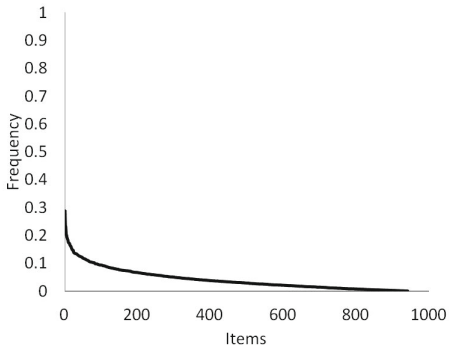

Fig. 4. Normalized item frequency distribution

In Table 4 we show the difference in itemsets generated and the time it took to generate items of varying minFreqSup on the T40I10D100K dataset. As we increase the minFreqSup, the number of itemsets generated increases and the relative time also increases. It is important to note that the real time taken for SRP-Tree to generate itemsets decreases as we increase the minFreqSup. This is because at a lower minFreqSup for this particular distribution of the dataset, there is a high co-occurrence of rare items to other items with similar item frequency (indicating that these other items are also likely to have rare properties). The mining of these additional highly connected items with rare association patterns incurred a larger overhead in maintaining the Connection Table. As the minFreqSup increases and most of the highly connected potential rare items are accounted for, the overhead decreases and the results in a faster runtime.

Table 5 shows the difference in execution time when the block size varies. The execution time is increased as the block size increases due to the increased size of the tree being built.

Table 4. Varying MinFreqSup for T40I10D100K

MinFreqSup	Itemset	SRP-Tree		SC-Tree	
		Time(s)	Rel. Time	Time(s)	Rel. Time
0.04	4397517	333	1.00	357	1.07
0.05	4734806	301	1.00	380	1.26
0.06	5028947	278	1.00	421	1.51
0.1	5105892	246	1.00	446	1.81
0.15	5136904	238	1.00	454	1.91

Table 5. Execution Time based on Varying Block Sizes for T40I10D100K

Block size	SRP-Tree		SC-Tree	
	Avg Time / Window (s)	Relative Time	Avg. Time / Window (s)	Relative Time
10K	41	1.00	70	1.70
25K	75	1.00	95	1.26
50K	142	1.00	201	1.42

6 Conclusions and Future Work

We present a new method for mining rare patterns using a tree structure in a data stream environment. To the extent of our knowledge, this is the first algorithm that looks at mining rare patterns in a data stream. Our technique is a one-pass only strategy which is capable of mining rare patterns in a static database or in a dynamic data stream. In the case of mining data streams, our technique is also capable of mining at any given point of time in the stream and with different window and block sizes. One of the contributions of this algorithm is a novel approach using a Connection Table which keeps track of related items in a sliding window and reduces the mining space during itemset generation. In our evaluations on six different datasets, the SRP-Tree is capable of generating itemsets in a more efficient manner compared to the SC-Tree.

In the future we intend to investigate on dynamically adapting the minRareSup and minFreqSup thresholds on-the-fly because data streams are volatile and neither setting fixed thresholds for all data nor defining the thresholds prior based on distribution is deemed suitable. We will also look at the possibility of dynamically adjusting the window size to reflect the density of incoming data in the stream. For example, if the new transactions in the window contained uninteresting or duplicate itemsets and rules, we could (through varying the window size) decide not to mine until more interesting itemsets and rules are captured. It will also be interesting to investigate the limitations of the tree with respect to the different characteristics and intensity of the incoming data stream.

References

1. Adda, M., Wu, L., Feng, Y.: Rare itemset mining. In: Proceedings of the Sixth International Conference on Machine Learning and Applications, ICMLA 2007, pp. 73–80. IEEE Computer Society, Washington, DC (2007)

2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, pp. 487–499 (1994)
3. Cheng, J., Ke, Y., Ng, W.: Maintaining frequent closed itemsets over a sliding window. J. Intell. Inf. Syst. 31, 191–215 (2008)
4. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proceedings of the Fourth IEEE International Conference on Data Mining, ICDM 2004, pp. 59–66. IEEE Computer Society, Washington, DC (2004)
5. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. Knowl. Inf. Syst. 10, 265–294 (2006)
6. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows (extended abstract). In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2002, pp. 635–644. Society for Industrial and Applied Mathematics, Philadelphia (2002)
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD 2000, pp. 1–12. ACM, New York (2000)
8. Koh, Y.S., Rountree, N.: Finding Sporadic Rules Using Apriori-Inverse. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 97–106. Springer, Heidelberg (2005)
9. Lee, C.H., Lin, C.R., Chen, M.S.: Sliding window filtering: an efficient method for incremental mining on a time-variant database. Information Systems 30(3), 227–244 (2005)
10. Leung, C.K.S., Khan, Q.I.: Dstree: A tree structure for the mining of frequent sets from data streams. In: Proceedings of the Sixth International Conference on Data Mining, ICDM 2006, pp. 928–932. IEEE Computer Society, Washington, DC (2006)
11. Li, H.F., Lee, S.Y.: Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Syst. Appl. 36, 1466–1477 (2009)
12. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 337–341 (1999)
13. Mozafari, B., Thakkar, H., Zaniolo, C.: Verifying and mining frequent patterns from large windows over data streams. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 179–188. IEEE Computer Society, Washington, DC (2008), http://dl.acm.org/citation.cfm?id=1546682.1547157
14. Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007, vol. 1, pp. 305–312. IEEE Computer Society, Washington, DC (2007)
15. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Sliding window-based frequent pattern mining over data streams. Information Sciences 179(22), 3843–3865 (2009)
16. Troiano, L., Scibelli, G., Birtolo, C.: A fast algorithm for mining rare itemsets. In: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, pp. 1149–1155. IEEE Computer Society, Washington, DC (2009)
17. Tsang, S., Koh, Y.S., Dobbie, G.: RP-Tree: Rare Pattern Tree Mining. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 277–288. Springer, Heidelberg (2011)