This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30044597

Reproduced with the kind permissions of the copyright owner.

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright : 2007, IEEE
Infrequent Item Mining in Multiple Data Streams

Budhaditya Saha, Mihai Lazarescu, Svetha Venkatesh
Institute for MultiSensor Processing and Content Analysis
Curtin University of Technology, Perth, Western Australia

Abstract
The problem of extracting infrequent patterns from streams and building associations between these patterns is becoming increasingly relevant today as many events of interest such as attacks in network data or unusual stories in news data occur rarely. The complexity of the problem is compounded when a system is required to deal with data from multiple streams. To address these problems, we present a framework that combines the time based association mining with a pyramidal structure that allows a rolling analysis of the stream and maintains a synopsis of the data without requiring increasing memory resources. We apply the algorithms and show the usefulness of the techniques.

1 Introduction
Infrequent pattern mining is concerned with extracting "rare" or "unusual" patterns from streams of data. In the past, frequent pattern mining has been investigated in detail with little research being done in infrequent pattern mining. However, infrequent patterns are often more useful than frequent patterns as they provide information about events of interest (such as in network intrusion detection).

In this paper we describe an approach which combines pyramidal trees with association rule mining to discover infrequent patterns in data streams as well as any associations between infrequent patterns across multiple data streams. The problem of effective storage of patterns has been investigated previously by Aggarwal et. al. [1] and Giannella et al. [4]. The work described used a logarithmic sized synopsis to store critical stream information. However, the work was focused on frequent patterns irrespective of their relevance, it did not handle varying time intervals and was aimed at single streams. Previously, Ma [5] has described an algorithm for mining infrequent but mutually dependent item sets but the work was only applicable to static databases rather than data streams.

We address two major issues in infrequent pattern mining: scalability in terms of memory requirements and pattern selection over time horizons that vary in span. We have developed a framework for identifying infrequent patterns in stream data which allows the discovery of the patterns at different time resolutions. The framework selects the infrequent patterns and stores them into a data structure such that only the unique patterns across the stream are stored at the top of the structure. An important aspect of our framework is that it can handle multiple data streams and thus allows the discovery of patterns that are infrequent across multiple data streams. The structure allows the incremental processing of the data points observed and it extends the horizon over which the data is processed automatically. The novelty and contribution of our research is that we present a framework to efficiently store and compare the patterns extracted from a data stream/s without requiring increasing amounts of memory resources for the processing/storing of the data. The paper is organized as follows: Section 2 covers previous related work, Section 3 describes the algorithm while Section 4 presents the results. Section 5 contains the conclusions.

2 Related Work
Stream processing[2] has created a lot of interest among the statistics and data mining community. An aspect of data mining that has attracted a large amount of attention lately has been in the area of mining “interesting” patterns from streams. Xin et.al. [8] has proposed a framework in which patterns are discovered according to the user’s preferences. However, the work makes the assumption that “interesting” patterns are found only by mining frequent patterns which is not valid in all applications. There has also been work done in discovering frequent patterns across distributed data streams. Manjhi [6] proposed finding out global iceberg queries over distributed bags using local icebergs. The method requires a multilevel hierarchical communications framework which is not well suited for stream data. Zhao [9] has described a sampling based load shedding technique for finding out a global iceberg. Babcock [3] described a method to monitor the top-k queries over distributed data streams which support both sum and average queries. Scal-
ability [7] is also a key aspect of our work. Aggarwal [1] has described an efficient method to store the synopsis of a continuous stream in a hierarchical fashion which resembles a pyramid. The information is stored in terms of cluster feature vectors, where the feature vectors are stored at each level of the pyramid. Each level of the pyramid represents a different granularity of time over a full time horizon.

There has been relatively little work done on infrequent pattern discovery from streams. Zhou [10] has recently described a process for identifying infrequent structures in XML documents. However, the algorithm is not suitable for streams as it requires multiple scans of the data.

3 Infrequent Pattern Mining Algorithm

Infrequent Pattern Processing Challenges: The problem of mining infrequent patterns and building associations from these patterns extracted from a stream poses two challenges: (1) the problem of extracting the patterns and storing them efficiently and (2) the problem that the infrequent patterns and associations discovered may over time become frequent. We are only interested in the patterns that remain infrequent over the entire stream (or part of) processed.

There are many techniques that can be used to extract infrequent patterns but typically, these techniques produce a large amount of infrequent items which require increasing amounts of resources both in terms of memory and computations. The typical approach [10] to infrequent pattern mining is to first identify the frequent patterns and then prune these patterns from the dataset. The remaining patterns are considered to be infrequent. The main problem is that a large number of infrequent items are typically generated by the extraction process and hence as more data is observed in the stream, more patterns need to be stored.

Preliminaries: Let $S$ be the data stream $[(d_1, i_1, t_1), (d_2, i_2, t_2), \ldots, (d_n, i_n, t_n)]$ where $(d_k, i_k, t_k)$ represents the data instance, its class label or item id and its timestamp in the stream respectively. Let $l$ represent the set of all class labels and thus $i_k \in l$. Since we are proposing a hierarchical data structure, let $h$ denote the height of the hierarchy, where the root node is at the level $l=0$ and the leaf node is at the level $l=H$. The data instances are processed at the leaf node level and the summary statistics are maintained at higher levels in the hierarchy. Since the stream is processed by a moving window of length $L$, let $u_l^l$ denote the $i^{th}$ window at level $l$ and its time span be denoted by $t^l_{s_l} : t^l_{e_l}$ and $t_{s_l}$ and $t_{e_l}$ are the start and end time span of $u_l^l$. Let $x(t^l_{s_l}, e_l)$ represent the set of items, $x^l(t^l_{s_l}, e_l)$ represent the item with id $k$, $x^l_k(t^l_{s_l}, e_l)$ represent the support of the item with id $k$ and $x_{a}(t^l_{s_l}, e_l)$ denote the set of mutually dependent infrequent items in $u_l^l$ over time span $t^l_{s_l} : t^l_{e_l}$ respectively. If item $i_k$ occurs $n_k$ times in $t^l_{s_l} : t^l_{e_l}$, then the support threshold ($\sigma_a = n_k/n$).

Given $\mu$ is the minimum support threshold for an item $i_k$, if $x^l_k(t^l_{s_l}, e_l) \geq \mu$, then $i_k$ is considered to be a frequent item, otherwise $i_k$ is considered to be an infrequent item. In an analogous way to the definition above $y(t^l_{s_l}, e_l)$, $y^l_k(t^l_{s_l}, e_l)$ and $y_{a}(t^l_{s_l}, e_l)$ represent the set of infrequent items, the infrequent item with item id $k$ and the support of the infrequent item $i_k$ in the window $u_l^l$ over the time span $t^l_{s_l} : t^l_{e_l}$ respectively. Hence, we can write $y_k^l(t^l_{s_l}, e_l) = \{ x_k^l(t^l_{s_l}, e_l) : x_k^l(t^l_{s_l}, e_l) < \mu \}$, $y_{a}(t^l_{s_l}, e_l) = \{ x_{a}(t^l_{s_l}, e_l) : x_{a}(t^l_{s_l}, e_l) < \mu \}$ and $y(t^l_{s_l}, e_l) = \{ y_k^l(t^l_{s_l}, e_l) \}, \forall k$. The summary statistics of two windows $(u^l_{j_1}, u^l_{j_2})$ and $(u^l_{j_2}, u^l_{j_3})$ are computed as follows:

$$\text{Summary}(t^l_{s_l}, e_l) = y(t^l_{s_l}, e_l) if (y_k^l(t^l_{s_l}, e_l) < \mu, \forall k.$$}

Based on Ma’s [5] work on significantly mutually dependent item sets on transactional databases, we derived the following definition for mutually dependent infrequent items over a time span $t^l_{s_l} : t^l_{e_l}$: if $i_k$ and $i_j$ are two infrequent items (with $x_k^l(t^l_{s_l}, e_l) < \mu$ and $x_j^l(t^l_{s_l}, e_l) < \mu$), $(i_k + i_j)$ is the union of $i_k$ and $i_j$, support of $(i_k + i_j)$ is denoted by $x_{k+j}^l(t^l_{s_l}, e_l)$ and $\text{minp}$ represents the threshold for minimum probability of mutual dependence of items, then $i_k$ and $i_j$ are mutually dependent if $x_{k+j}^l(t^l_{s_l}, e_l) \geq \text{minp}$ and $x_{k+j}^l(t^l_{s_l}, e_l) \geq \text{minp}$. If the above two conditions are satisfied then the mutually dependent items over time span $t^l_{s_l}$ are defined as follows:

$$x_{a}(t^l_{s_l}, e_l) = \{ i_k + i_j \}.$$}

Finally, let $\tau$ be the threshold that indicates whether or not an infrequent pattern $x^l_k(t^l_{s_j}, e_k)$ is noise. Noise, unlike infrequent patterns, is random and lacks both consistency and persistence. It has very low support and would not be part of any item temporal associations. Hence if support $x_k^l(t^l_{s_j}, e_k) \leq \tau$ over time span $T_{j_k}$ where $t_j \ll t_k$ and $\tau = \mu/10$, then $x_k^l(t^l_{s_j}, e_k)$ is considered to be noise.

Algorithm Overview: We process the data in two stages: first we eliminate the frequent items and second we build the associations between the infrequent items. Overall, our framework has three stages. First, the initial set of infrequent patterns are extracted and stored in the pyramid. Second, the rest of the stream is processed and the infrequent pattern set updated. In the final stage the infrequent pattern set for the entire stream is finalised and compared with the sets extracted from other streams in order to determine the pattern set that is infrequent across streams.

3.1 Extracting the Infrequent Patterns from the Data Stream

Entropy Based Window Selection: To extract the initial set of infrequent patterns from the stream, the algorithm first selects windows that are good candidates for a detailed infrequent item analysis. The candidates are selected by computing the entropy of the window because the entropy measure determines the uniformity of the items in the win-
dows. By choosing the proper entropy coefficient $\eta$, we filter out most frequent items as early as possible. To compute the entropy in a data window, we divided the data stream into intervals of length $L$ and used the following definition. Given $p[x^k(t^0[s_i, e_i])]$ is the probability of item $i$ in window $t^0$. The entropy of the window $w^0_t$ is defined as:

$$E[w^0_t] = -\sum_i p[x^k(t^0[s_i, e_i])] \log_2 p[x^k(t^0[s_i, e_i])]$$  (3)

In our case, if $E[w^0_t] > \eta$, then the window was considered to be a candidate for pattern mining.

**Infrequent Pattern and Association Extraction:** Once a candidate window has been identified, we extract all infrequent patterns $y(t^0[s_i, e_i])$ for which $\lambda_{\text{in}_\text{frequent}}$. The processing is repeated for all windows in the candidate queue thus producing the infrequent sets for the individual windows. However, this process only ensures that the patterns are infrequent for each window rather than across multiple windows or across parts of the entire stream. To ensure that the patterns are infrequent across more than one window, it is necessary to compare the support of the infrequent items sets across windows. This requires that we update the support for each infrequent item as more windows are processed and involves the summary of two consecutive candidate windows. The summary of two windows consists of the items that are “infrequent” in both windows. If there are infrequent items similar in both windows and the sum of their support exceeds the threshold limit $\mu$, then these items would not be included in the summary of the windows.

**3.2 Storing the Infrequent Patterns**

The next step in the processing involves building the pyramidal data structure that stores the infrequent patterns discovered from the stream. We use a pyramidal framework similar to that described by Aggarwal [1]. The pyramidal data structure is built by merging the infrequent patterns sets extracted from increasingly larger ordered groupings of candidate windows. Consider the example in Figure 1. The bottom level contains all the infrequent item sets covered by a predefined time span (T) which in the case of Figure 1 has 8 candidate windows. The nodes at the next level in the pyramid contain the set of items that are infrequent for pairings of candidate windows. As the item sets are generated at the higher levels in the pyramid, the algorithm checks to ensure that items are indeed infrequent by checking the item against all the data points contained in the windows associated with that branch of the pyramid.

Consider the following example. Let $y^k(t^0[s_1, e_1])$ and $y^{k+4}(t^0[s_1, e_1])$ be patterns that are infrequent at level 0 in window $w^0_t$ in Figure 1. These two patterns would be candidates to be propagated to the level 1 node that summarizes the infrequent patterns from windows $w^0_0$ and $w^0_2$. However, before the patterns are propagated to the node at level 1, the patterns in window $w^2_0$ also have to be analyzed. Hence, as the data in window $w^0_2$ is processed, if the support for either $y^k(t^0[s_1, e_1])$ or $y^{k+4}(t^0[s_1, e_1])$ increases to a value that exceeds $\mu$, then the pattern is no longer propagated to the node at level 1. Therefore, an infrequent item at Level 2 in the pyramid that was extracted originally from window $w^0_0$ would be checked against the data from windows $w^0_0$, $w^0_2$ and $w^0_4$. Similarly, infrequent items extracted from the windows $w^0_0$ and $w^0_2$ at Level 1 would be checked against windows $w^0_2$, $w^0_0$. This process removes the dependency on the length of the window and for any value of $L$, we will always obtain the same infrequent items at the root of the pyramidal tree as outlined below.

**Lemma 1:** The infrequent patterns extracted from each window are independent of the length of the data window.

**Proof:** If the length of the moving window ($L$) = $t^0[s_1, e_4]$ (see. Figure 2), then the infrequent items over the time span $t^0[s_1, e_4]$ derived using equation 1 are given by $y(t^0[s_1, e_4]) = \text{Summary}(t^0[s_1, e_4])$. If we change the length of the window from $L$ to $L_2$, then the time span would be covered by two equally sized windows $L'$ and $L''$ where $L' = t^0[s_1, e_2]$ and $L'' = t^0[s_3, e_4]$. Moreover, we can write that $t^0[s_1, e_4] = t^0[s_1, e_2] + t^0[s_3, e_4] = t^0[s_1, e_1] + t^0[s_2, e_2] = t^0[s_1, e_1]$. In addition, by using equation 1 we obtain $y(t^0[s_1, e_1])$ from the union of $y(t^0[s_1, e_1])$ and $y(t^0[s_2, e_2])$. Since, $y(t^0[s_1, e_1])$ covers all the infrequent items over the time span $t^0[s_1, e_4]$, then $y(t^0[s_1, e_1]) = y(t^0[s_1, e_4])$. Similarly, if we change the length of the window to $L/4$, then using equation 1 we can mine all infrequent items over the same time span. Therefore, extracting the infrequent items over a time span is independent on the length of the moving window.

**Lemma 2:** Pruning old data windows does not affect the infrequent patterns stored at the root of the pyramidal tree.

**Proof:** Given a sequence of $m$ processed candidate windows, the pyramid derived the windows will at the root
level \((l = h)\) store the infrequent patterns defined by equation 1 i.e. 
\[ y(l_i^S[s_m/2, e_m/2]) = \text{Summary}(l_i^S[s_1, e_m]) \]
that cover the time span \(l_i^S[s_1, e_m]\) (i.e. \(l_i^S[s_m/2, e_m/2]\) = \(l_i^S[s_1, e_m]\) as shown in Figure 3). Because of the properties of the infrequent items extracted using equation 1 which tracks the time interval and the support for each item, for any node in the pyramid at level \(l\), we can prune any of the \(l-1\) or lower nodes/branches from the pyramid without affecting the infrequent patterns stored in the nodes at level \(l\).

**Pruning Old Data:** The pruning process involves selected forgetting. We remove the oldest item sets (windows) and "recent" frequent patterns by pruning each branch of the existing tree and adding up new item sets (windows) as shown in Figure 4. This approach allows the algorithm to maintain the height of the tree and the computational resources for processing data points within a predefined limit.

To do this automatically, we process the data stream using a damped window model. We assign a weight \((m_i)\) to every window \((w_i)\) and the total weight of the candidate windows over time span \(T\) is decreased exponentially as a function of time \(f(t) = e^{-\delta t}\) where \(\delta \geq 0\). Given that \(N\) is the number of windows seen per unit time, \(\Delta\) is the time needed to process each window of length \(L = (\frac{T}{m})\), \(W_m\) is the total weight of the windows and \(\delta\) is the damping coefficient then the weight associated with each node \(n_{l=0}(w_i)\).

The total weight of a candidate window over the time span \(T\) is defined as \(W_m = \sum m_i e^{-\delta(T_{0..i})} + m_i e^{-\delta(T_{1..2})} + \cdots + m_i e^{-\delta(T_{m-1..m})}\). At initial phase i.e when \(t=0\), let \(m_1 = m_2 = \cdots = 1\), and \(T_{0..1} = \Delta, T_{1..2} = 2\Delta, \cdots, T_{m-1..m} = m\Delta\), then the equation becomes:

\[
W_m = \frac{1 - e^{-\delta m \Delta}}{1 - e^{-\delta \Delta}}
\]  
(4)

Moreover, if \(T = m\Delta \rightarrow \infty\), then,

\[
W_m = w_{minimum} = \frac{1}{1 - e^{-\delta \Delta}}
\]  
(5)

where, \(\delta\) is a constant and \(\Delta\) is dependent on the length of the window. For a given window size \(w_{minimum}\) is constant and if the total weight of the window decreases with time, then we prune the oldest windows and include new windows to keep \(w \geq w_{minimum}\).

**3.3 Infrequent Pattern and Rule Mining Across Streams**

The last stage in our framework expands the processing outlined for an individual stream to multiple streams. Given the streams \(S_{1...n}\), we compare each pattern stored at the root level in the pyramidal tree of stream \(S_r\) with the patterns stored at the root level of the pyramidal trees generated for the other streams (as shown in Figure 5). Similar to the summarization and update process described earlier in this section, we keep track of the cumulative support for each pattern and remove all those that exceed the predefined threshold \(\mu\). The remaining patterns in \(S_r\) that satisfy the condition that \(x_r(t^S[s_n/2, e_n/2]) < \mu\) are considered to be infrequent across all streams \(S_{1...n}\).

**4 Experimental Evaluation**

To validate our framework, we conducted a series of experiments using two real-world datasets: the KDD Cup 1999 network intrusion detection dataset and a new data set collected from the RSS feed over a period of two weeks.

**4.1 KDD’99 Cup Dataset**

The KDD’99 Cup intrusion dataset has around 4.9 million network connection records. Each connection record in the dataset is labeled as either normal or as a specific attack (24 types of attacks). We used sampling to divide the original dataset into ten streams of roughly 498,000 records each. The streams covered only a subset of the attacks in the original dataset and therefore for each stream we recorded the label and the frequency of the attacks contained in the stream. In all experiments, we used an entropy value of \(\eta = 0.4\) to determine whether a data window is used in the infrequent pattern extraction stage in the algorithm.

**Window Size Analysis:** The aim of the first set of experiments is to determine whether the same set of infrequent patterns is propagated to the root of the pyramidal data structure that covers an entire stream, regardless of the window size used to process the data. Column 2 in Table 1 contains the infrequent attack patterns for each stream and was used as the ground truth in the evaluation. Please note that **not** all attack patterns were infrequent - specifically, attacks 1 to 4 occurred frequently in most streams when compared with attacks 5 to 24. We processed the streams using a varying window size \(w\) that covered an interval of points ranging from as few as 2000 data points to the entire stream. The results are shown in Table 1 of the appendix. Of significance are the columns showing the difference \(\Delta\) between the ground truth and the infrequent patterns stored at the root of the pyramid. The results demonstrate that the changes made to the window size had no effect on the final set of infrequent items.

**Infrequent Pattern Frequence Analysis:** The aim of the second type of experiments was to determine how the frequency of patterns affects the level to which an infrequent pattern is propagated up in the pyramidal tree. We fixed the...
Table 1. Infrequent item sets extracted for each data stream processed with varying window sizes.

<table>
<thead>
<tr>
<th>Stream (S)</th>
<th>Level 2 Pattern Set</th>
<th>Ground Truth Set</th>
<th>Level 3 Pattern Set</th>
<th>Ground Truth Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>5, 8, 9, 12, 13, 15</td>
<td>5, 8, 9, 12, 13, 15</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$S_2$</td>
<td>6, 10, 11, 15, 18, 20</td>
<td>6, 10, 11, 15, 18, 20</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$S_3$</td>
<td>7, 8, 10, 11, 12, 14, 15, 17, 22</td>
<td>7, 8, 10, 11, 12, 14, 15, 17, 22</td>
<td>12, 17</td>
<td>5</td>
</tr>
<tr>
<td>$S_6$</td>
<td>5, 12, 14, 17, 18, 19, 22, 23</td>
<td>5, 12, 14, 17, 18, 19, 22, 23</td>
<td>5, 22</td>
<td>5, 22</td>
</tr>
</tbody>
</table>

Table 2. Infrequent Items Found at Levels 2 and 3 in the Pyramidal Tree.

<table>
<thead>
<tr>
<th>Stream (S)</th>
<th>Infrequent Item Set</th>
<th>Mutually Dependent Items (Temporal Association Rules)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>5, 7, 11, 13, 14, 16, 20</td>
<td>5 ⇔ 11, 13 ⇔ 20, 14 ⇔ 16</td>
</tr>
<tr>
<td>$S_2$</td>
<td>6, 12, 20, 21, 24</td>
<td>20 ⇔ 21</td>
</tr>
<tr>
<td>$S_7$</td>
<td>13, 17, 18</td>
<td>13 ⇔ 17, 13 ⇔ 18</td>
</tr>
</tbody>
</table>

Table 3. Mutually Dependent Items (Temporal Association Rules) generated from the items at the root of the pyramidal tree.

<table>
<thead>
<tr>
<th>Stream (in Days)</th>
<th>Infrequent Items in the Stream</th>
<th>Infrequent Items Obtained</th>
<th>Difference $\Delta$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day-1</td>
<td>63</td>
<td>63</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-2</td>
<td>66</td>
<td>65</td>
<td>0.01</td>
</tr>
<tr>
<td>Day-3</td>
<td>66</td>
<td>66</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-4</td>
<td>108</td>
<td>110</td>
<td>-0.01</td>
</tr>
<tr>
<td>Day-5</td>
<td>126</td>
<td>123</td>
<td>0.02</td>
</tr>
<tr>
<td>Day-6</td>
<td>104</td>
<td>104</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-7</td>
<td>104</td>
<td>104</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-8</td>
<td>63</td>
<td>66</td>
<td>-0.04</td>
</tr>
<tr>
<td>Day-9</td>
<td>69</td>
<td>68</td>
<td>0.01</td>
</tr>
<tr>
<td>Day-10</td>
<td>72</td>
<td>75</td>
<td>-0.04</td>
</tr>
<tr>
<td>Day-11</td>
<td>144</td>
<td>145</td>
<td>0.01</td>
</tr>
<tr>
<td>Day-12</td>
<td>150</td>
<td>150</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-13</td>
<td>201</td>
<td>201</td>
<td>0.0</td>
</tr>
<tr>
<td>Day-14</td>
<td>192</td>
<td>190</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 4. The comparison between the infrequent items obtained from the pyramidal tree and the ground truth for the RSS dataset.
depth of the pyramidal tree to a depth(D) = 3 with the middle level nodes covering the infrequent items extracted from 4 windows of size w. Items with support < μ for i = 1..4w would be expected to reach the middle level nodes of the pyramidal tree, while items with support < μ for i = 1..nw would be found at the top level of the pyramid. We used a window size w = 2000 to process the data stream and we recorded the infrequent patterns extracted at all levels in the pyramid along with the window id. The summarized results for streams 3, 4, 5 and 6 are shown in Table 2.

Multiple Stream Infrequent Pattern and Association Rule Mining: The aim of the last experiment was to derive temporal associations from the infrequent item sets discovered in the streams. At the root level only six of the ten streams had multiple infrequent items and furthermore, only three of these streams contained enough support to generate associations between the infrequent items. The associations derived using a window size of 2000 data points are shown in Table 3. The middle column in the table shows the infrequent item set stored at the root of the pyramidal tree while the last column shows the associations that were derived from the stream data. For both streams S1 and S2 only a subset of the infrequent item set was used in the rules. This was due to the lack of support for rules that would involve the remaining infrequent items. The rules indicate temporal associations between the infrequent items. For example, the rule 5 → 11 indicates that whenever an attack of type 5 was observed, one can also expect an attack of type 11 within 2000 data points.

4.2 RSS Dataset

The second dataset consisted of news stories collected based on a RSS feed. Using the RSS feed, we collected stories every hour for a period of two weeks for two topics: sports news stories and financial news stories. The data was pre-processed using WORDNET. After the pre-processing step, the RSS data was divided into 14 streams with each stream consisting of the ordered sequence of stories collected over a period of 24 hours. All the data was manually checked to extract the ground truth.

Infrequent News Story Mining: The aim of the experiment was to identify stories that are infrequent over a period of 24 hours. Stories that would start as infrequent were expected to contain words which would not be repeated frequently in the subsequent stories. The RSS data contained a significant amount of infrequent patterns and associations and cannot be reproduced here in detail. We provide the summary statistics collected from the 14 streams in Table 4 and compare the results with the ground truth. Columns 2 shows the number of infrequent patterns extracted by our approach while column 3 shows the ground truth. The difference is shown in column 4. Please note that unlike the case of the KDD’99 data, column 4 shows that there was a difference between the infrequent patterns and associations stored in the root node of the pyramidal tree and the ground truth. The reason for the difference is that some of the infrequent patterns discovered towards the end of the streams could not be propagated to the root node as there was not enough data to allow this.

5 Conclusions

The framework proposed in this paper can be used effectively to extract infrequent items from stream data and generate temporally ordered associations (mutually dependent items) from these items. The results obtained have demonstrated that the algorithm can handle vastly different types of data. The major advantages of our work are that it allows incremental processing of data streams with limited memory resources and it can be applied to multiple streams.

References