Mining Frequent Patterns Across Multiple Data Streams

Jing Guo⋆†, Peng Zhang⋆, Jianlong Tan⋆, Li Guo⋆

⋆ Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
† Beijing University of Posts and Telecommunications, Beijing, 100876, China
guojing@software.ict.ac.cn, {zhangpeng, tjl, guoli}@ict.ac.cn,

ABSTRACT
Mining frequent patterns from data streams has drawn increasing attention in recent years. However, previous mining algorithms were all focused on a single data stream. In many emerging applications, it is of critical importance to combine multiple data streams for analysis. For example, in real-time news topic analysis, it is necessary to combine multiple news report streams from different media sources to discover collaborative frequent patterns which are reported frequently in all media, and comparative frequent patterns which are reported more frequently in a media than others. To address this problem, we propose a novel frequent pattern mining algorithm Hybrid-Streaming, H-Stream for short. H-Stream builds a new Hybrid-Frequent tree to maintain historical frequent and potential frequent itemsets from all data streams, and incrementally updates these itemsets for efficient collaborative and comparative pattern mining. Theoretical and empirical studies demonstrate the utility of the proposed method.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications—Data Mining

General Terms
Algorithm

Keywords
Multiple data streams, Data stream mining, Frequent pattern mining

1. INTRODUCTION
Data stream mining has drawn increasing attention in recent years [1, 2, 3, 4, 5]. Mining frequent patterns from data streams represents one of the most important directions in data stream mining community [1]. Due to the rapid growth of data volumes, frequent pattern mining on data streams needs to mine from large volumes of continuous stream data in limited memory. To this end, many data stream frequent pattern mining algorithms have been proposed recently, such as the FP-Stream algorithm [1], Lossy Counting algorithm [2], Count Sketch algorithm [3], to name a few. Although these algorithms have demonstrated their effectiveness, a limitation is that they were all designed for mining a single data stream. In many emerging applications, stream data are often generated and collected from multiple sources, and it is necessary to combine multiple data streams for mining.

Example 1. Let us take a real-time news topic analysis system for example. The system collects news report streams from three public social media: BBC, New York Times, and The Globe And Mail. The essential goal of the system is to analyze some interesting and insightful patterns. Specifically, the following four tasks need to be addressed:

(1) Which topics were reported frequently by all the three media in the last 12 hours?
(2) Which topics were reported frequently by The Globe And Mail, while infrequently by BBC and New York Times in the last 12 hours?
(3) Which topics were not only reported frequently by The Globe And Mail in the last 12 hours, but were also reported frequently by BBC and New York Times in the last 24 hours?
(4) Which topics were reported frequently by all the three media, while more frequently by BBC in the last 12 hours?

Tasks (1) and (3) are defined as mining collaborative frequent patterns, as they aim to find the common frequent patterns across multiple data streams. Tasks (2) and (4) are defined as mining comparative frequent patterns, as they aim to find contradicting frequent patterns across multiple data streams.

In order to solve these tasks, it is necessary to combine multiple data streams for analysis. Combining multiple data streams for mining has been studied previously. For example, clustering multiple streams [6], classifying multiple data streams [7], analyzing correlations among multiple streams [8], and privacy preserving mining across multiple data streams [9]. However, none of them considers the problem of mining collaborative and comparative frequent patterns across multiple data streams. On the other hand, although some previous work [10] considered mining frequent patterns across multiple databases, the method cannot be directly used in data streams.

In order to discover frequent patterns across multiple data streams, an intuitive method is to maintain and update the frequencies of all historical stream data into a summary structure, based on which all the collaborative and comparative patterns can be retrieved accordingly. However, with this method, the following concerns need to be addressed:

(1) How to efficiently maintain historical frequent itemsets from all data streams? A simple solution is to extend the single data
stream frequent mining structure (e.g., FP-Stream structure [1]) to multiple data streams. In other words, each data stream can be maintained in an independent FP-Stream structure, and the results from each FP-Stream can be combined for mining. Although this method is very easy to implement, a limitation is that its memory cost increases linearly with the number of streams. For example, in order to mine from one hundred data streams, we have to build one hundred FP-Stream structures. This is unaffordable for many applications. A better alternative method is to construct a single global data structure to make full use of shared itemsets across all the data streams.

(2) How to update historical frequent itemsets? In data streams, an infrequent itemset has the potential to grow into a frequent itemset. Thus, we cannot delete an infrequent itemset at each given time point. On the other hand, due to the huge number of infrequent itemsets in data streams, it is impractical to maintain them all in memory during mining. Thus, it is necessary to find a compromise solution between accuracy and memory cost.

(3) Based on the historical itemsets, how to design an effective method to discover collaborative and comparative patterns across all streams. The algorithm should be able to discover these complex patterns not only from the same time interval, but also from different time intervals.

In summary, in order to discover collaborative and comparative frequent patterns across multiple data streams, the following concerns need to be addressed:

- How to design an efficient data structure to maintain historical frequent itemsets from all streams?
- How to design an approximate updating algorithm to efficiently update the frequent itemsets from all streams?
- How to retrieve collaborative and comparative frequent patterns from the designed data structure?

To address these challenges, a novel Hybrid-Stream algorithm (H-Stream for short) is proposed to discover collaborative and comparative frequent patterns across multiple data streams. H-Stream first employs a new Hybrid-Frequent Tree (H-tree for short) to maintain historical frequent and potential frequent itemsets for all streams, and then it incrementally updates these itemsets under the Chernoff bound. Based on the maintained itemsets in H-tree, the collaborative and comparative patterns can be efficiently retrieved by traversing the tree. Theoretical and empirical studies demonstrate the utility of the proposed method.

The rest of this paper is organized as follows. Section 2 introduces the H-Stream algorithm and analyzes its performance. Section 3 reports experimental results and comparisons. We conclude the paper in Section 4.

## 2. THE H-STREAM ALGORITHM

In this section, we first introduce the H-tree structure in Section 2.1, based on which we design the H-Stream algorithm in Section 2.2. We analyze the performance of the algorithm in Section 2.3.

### 2.1 The H-tree structure

Structurally, H-tree consists of three components:

1. A Tree (T) that maintains frequent and potential frequent itemsets discovered from all streams. Itemsets in data streams are categorized into three types: frequent itemset whose current support is larger than \(\delta\); infrequent itemset whose current support is lower than \(\delta - \epsilon\), where \(\epsilon\) is the error bound; and potential frequent itemset whose current support lies between parameters \(\delta\) and \(\delta - \epsilon\). We will discuss this parameter later in Section 2.3.

Ideally, all these three types of itemsets can be maintained for accurate mining. But due to the limited memory, only the frequent and potential frequent itemsets will be maintained in the tree.

2. A Hybrid-window \(W\) that records the frequencies of the maintained itemsets in each time interval. Moreover, to further reduce the memory cost of H-tree, a logarithmic tilted window [1] can be used here.

3. A Header table \(H\) that consists of an array of pointers that point to the itemsets in distinct level of tree. By using table \(H\), the frequent and potential frequent itemsets can be efficiently retrieved from tree \(T\). This will reduce the overall update time of H-tree.

Compared to maintain each data stream independently, H-Tree can achieve a more compacted memory cost by making full use of the overlapping structures among all maintained itemsets.

### 2.2 Algorithm Description

Algorithm 1 shows the whole procedure of the H-Stream algorithm. Obviously, H-Stream mainly consists of three steps. In the first step, it builds an H-tree structure to maintain the frequent and potential frequent itemsets from all streams. Note that the numbers of these itemsets are controlled by parameters \(\delta\) and \(\epsilon\). In the second step, H-Stream continuously updates H-tree by inserting new emerging frequent and potential frequent itemsets, meanwhile deleting itemsets that have become infrequent. In the last step, H-Stream traverses the tree to retrieve collaborative and comparative frequent patterns.

There are five inputs of Algorithm 1. Parameter \(\delta\) controls the number of frequent itemsets. The lower value of \(\delta\), the more frequent itemsets we can obtained. Parameter \(\epsilon\) controls the tree prune rate. The lower value of \(\epsilon\), the more infrequent itemsets will be deleted. Besides, we assume the \(m\) data streams \(S\) are processed batch-by-batch, and data from the \(j\)th stream \((1 \leq j \leq m)\) during time interval \(t_i\) \((i \geq 1)\) is denoted by \(B^j_i\). The collaborative parameter set \(\alpha\) and the comparative parameter set \(\beta\) vary case by case. They can be manually set by end users.

In the first step of Algorithm 1, H-Stream uses the first batch data \(B^j_1\) \((1 \leq j \leq m)\) to construct an H-tree. This can be further split into two sub-steps: (1) Discover frequent and potential frequent itemsets from \(B^j_1\). Specifically, frequent itemsets \(U\) \((\text{i.e., } \sup(U) \geq \delta)\), and potential frequent itemsets \(V\) \((\text{i.e., } \sup(V) \geq \delta - \epsilon)\) will be extracted from all \(B^j_1\) \((1 \leq j \leq m)\) using FP-growth algorithm [11]; (2) Initialize the three components of H-tree. Assume all items distribute uniformly in the streams, and all the distinct items in the first batch can be observed. We calculate the average frequencies for all the distinct items to get an item list in descending order. To
We study H-Stream’s performance from both accuracy loss and memory cost viewpoints. As we discussed above, H-Stream uses parameters (δ - ϵ) to prune H-tree. Thus, we analyze H-Stream’s performance under this prune method.

3. EXPERIMENTS

News Report Streams were collected from three public social media: BBC, The Globe And Mail, and New York Times from July 22 to July 29, 2011. The news titles span over politics, technology, entertainment, and sports. In our experiments, a news title is taken as a transaction.

Table 1: Parameters used in the IBM data generator.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>S_1</th>
<th>S_2</th>
<th>S_3</th>
<th>S_4</th>
<th>S_5</th>
<th>S_6</th>
<th>S_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>tlen</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>nitems</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>npats</td>
<td>20</td>
<td>22</td>
<td>25</td>
<td>23</td>
<td>25</td>
<td>24</td>
<td>21</td>
</tr>
</tbody>
</table>

The IBM Quest Market-Basket Synthetic Data Streams were generated by IBM Quest Market-Basket Synthetic Data Generator [12]. The parameters of the data generator are listed in Table 1. We generated seven data streams, each having a batch size of 10,000 transactions.

1The exact algorithm maintains all the transactions in data streams for mining, which is supposed to be the best algorithm in term of mining accuracy.
Benchmark Method We compare H-Stream with another Approximation Stream mining algorithm, A-Stream for short. A-Stream is a variation of H-Stream. Different from H-Stream that uses both $\epsilon$ and $\delta$ to prune H-tree, A-Stream uses only $\epsilon$ for pruning. In so doing, A-Stream can maintain more potential frequent itemsets.

Parameter Study on $\epsilon$ The error rate $\epsilon$ is an important parameter for pruning H-tree. The comparison results are shown in Table 2 and Fig. 2. From Table 2, we can observe that H-Stream and A-Stream perform almost equally well. From Fig. 2, we can observe that H-Stream maintains much fewer nodes than A-Stream. This also validates our claim in Theorem 1.

4. CONCLUSIONS

In this paper, we study a novel problem of mining collaborative and comparative frequent patterns across multiple data streams. To achieve this goal, we propose a new frequent pattern mining algorithm called H-Stream. H-Stream uses a new H-tree to maintain historical frequent and potential frequent itemsets, and incrementally updates these itemsets for efficient collaborative and comparative mining. Theoretical and empirical studies have demonstrated the utility of the proposed method.

5. ACKNOWLEDGMENTS

This research was supported by the National Science Foundation of China (NSFC) under Grant No. 61003167, Basic Research Program of China (973 Program) under Grant No. 2007CB311100.

6. REFERENCES

[1] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. AAAI/MIT, 2003.

Mining results from the real world news report streams Stream news received in one day will be taken as a batch. A keyword will be taken as a frequent item if it is reported no less than three times a day. Due to space limitation, we only summarize four interesting patterns as follows.

(1) The keyword Norway was reported frequently by all the three media from July 26 to July 27, 2011 (collaborative frequent pattern).
(2) The keyword Olympic was reported frequently by BBC, while infrequently by New York Times and The Globe And Mail from July 25 to July 29, 2011 (comparative frequent pattern).
(3) The keyword China was not only reported frequently by New York Times and BBC from July 26 to July 27, 2011, but was also reported frequently by The Globe And Mail from July 22 to July 25, 2011 (collaborative frequent pattern).
(4) The keyword debt was reported frequently by all the three media, while more frequently by New York Times from July 22 to July 29, 2011 (comparative frequent pattern).