Sequence Clustering in Data Streams

A. Marascu  F. Masseglia

AxiS Project-Team
INRIA – Sophia Antipolis

Goals:

• Extract sequential patterns from data streams. Applied to: behaviour of a Web site’s users.

• Identifying problems arising with this pattern extraction. Particularly the management of their history.

Framework: The SCDS method
(Sequence Clustering in Data Streams)
Data Streams: a few words…

- New elements are generated continuously.
- Data have to be considered as fast as possible.
- No blocking operator can be performed.
- Data can be examined only once.
- Memory usage is restricted.

Sequential Pattern Mining: some definitions.

- **Item**: bought by a customer
- **Transaction**: a customer + an item + a timestamp
- **Sequence**: ordered list of itemsets

- **Data sequence**: stands for the activities of a customer. Let \( T_1, T_2, \ldots, T_n \) be the transactions of \( C_j \), the data sequence of \( C_j \) is:
  \(< \text{itemset}(T_1) \ \text{itemset}(T_2) \ \ldots \ \text{itemset}(T_n) >\)

- **Minimum support**: the minimum number of occurrences of a sequential pattern to be considered as frequent.
Illustration:

| U1 | Publications | Paper1 | Paper2 | Paper3 |
| U2 | Publications | Paper1 | List   | Paper2 |
| U3 | Research     | Theme1 | Theme3 | Theme4 |
| U4 | Publications | List   | Paper1 | List   |
| U5 | Research     | Theme1 | Theme2 | Theme3 |

Question: «Can we find a behavior that would be shared by (at least) 40% of the users recorded in the log file?»

*behaviour*: a series a requests performed during a navigation on the site.

Extracting patterns from data streams

1) **Satisfy the constraints of a data stream environment.**
   - High speed algorithms.
   - Sampling with an estimation of the quality.
   - etc.

2) **Managing the history of frequencies**
   - Logarithmic Tilted Time Window (Han et al.)
   - Segment Tuning and Relaxation (Teng et al.)

Frequent itemset (a b c)

<table>
<thead>
<tr>
<th>time</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 min</td>
<td>0.17</td>
</tr>
<tr>
<td>15 min</td>
<td>0.18</td>
</tr>
<tr>
<td>30 min</td>
<td>0.25</td>
</tr>
<tr>
<td>1 hour</td>
<td>0.12</td>
</tr>
<tr>
<td>2 hours</td>
<td>0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256 days</td>
<td>0</td>
</tr>
</tbody>
</table>
Overview

Stream

Batch \( B_n \)
Batch \( B_{n-1} \)
Batch \( B_{n-2} \) …

Data mining

Result \( B_n \)
Result \( B_{n-1} \)
Result \( B_{n-2} \)

Example: FTPStream

Stream

Batch \( B_n \)
Batch \( B_{n-1} \)
Batch \( B_{n-2} \) …

FP-tree

FP-Growth

FP-tree

FP-stream

<table>
<thead>
<tr>
<th>Item window</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i_2 )</td>
<td>0.23</td>
</tr>
<tr>
<td>( i_2 )</td>
<td>0.30</td>
</tr>
<tr>
<td>( i_1 )</td>
<td>0.30</td>
</tr>
<tr>
<td>( i_2 )</td>
<td>0.30</td>
</tr>
</tbody>
</table>
Why is this such a deal to extract sequential patterns from a data stream?

A sequential pattern mining algorithm may be based on:

- Breadth-first search
- Depth-first search
- Without candidate generation
- Sampling

Size of the result!  
Size of the batch!
“We have to find the balance between the execution time and the quality of the extracted patterns.”

Our proposal relies on two compromises:

1. A greedy sequence clustering algorithm.
2. A sequence alignment method applied to each cluster.

---

A Greedy Algorithm for clustering streaming sequences

Builds the clusters on the fly.

Let \( S \) be the current sequence. Scan \( C \), the set of clusters. Let \( C_j \) be the cluster having the most similar centroid to \( S \).

- insert \((S, C_j)\);
- update_centroid \((C_j)\);

If no cluster has been found then create_cluster \((C_j, S)\);

When "n" sequences have been processed: next batch.
Sequence alignment for each cluster

The centroid is the result of an alignment applied to the sequences of each cluster:

\[
\begin{align*}
< (a) (b) (d) > & \Rightarrow < (a) (b) (d) > \\
< (a) (c) (d) > & \Rightarrow < (a) (c) (d) > \\
\end{align*}
\]

Filter \( k = 1 \): \(< (a:2) (b:1, c:1) (d:2) >\)
Filter \( k = 2 \): \(< (a:2) (d:2) >\)
Managing the history of the extracted patterns

- On static databases, the knowledge is stable
- On data streams, the knowledge is evolving with the stream

Ongoing work: an incremental clustering.

Motivation: The division of the stream into batches "blurs" the history of the frequent patterns.
We propose to perform an incremental clustering in order to maintain a coherent history.

- **Idea**: have the cluster evolving.
- **Objective**: be independent from slight variations when managing the history of extracted patterns.
- **Principle**: keep the centroid (aligned sequence) of the clusters from one batch to another.

<table>
<thead>
<tr>
<th>15 min</th>
<th>15 min</th>
<th>30 min</th>
<th>1h</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(BC)D</td>
<td>A(BC)D</td>
<td>A(BC)DE</td>
<td>A(BC)E</td>
<td>...</td>
</tr>
<tr>
<td>3%</td>
<td>4%</td>
<td>2%</td>
<td>2%</td>
<td>...</td>
</tr>
</tbody>
</table>

A few questions motivating this work:

- Is data mining able to help summarizing a stream?
- What should this summary look like?
- Where does the approximation stop?