RULEGROWTH: Mining Sequential Rules Common to Several Sequences by Pattern-Growth

Philippe Fournier-Viger
Roger Nkambou
Vincent Shin-Mu Tseng

1National Cheng Kung University (Taiwan)
2University of Québec in Montréal (Canada)

March 2011 - ACM SAC 2011 - Taichung
Introduction

• Many databases contain large amount of temporal information.

• It is a challenge to develop algorithms for discovering useful temporal patterns in these databases.

• Example:
  – to predict stock market fluctuations,
  – to understand and predict consumer behavior in online web stores,
  – to predict the effect of medicines on patients.

• In this paper, we are interested by sequence databases containing sequences of discrete events (e.g. protein sequences and clicks sequence on websites).
Sequence Database

• Each **sequence** is an time-ordered list of itemsets.

• An **itemset** is an unordered set of items (symbols), considered to occur simultaneously.

<table>
<thead>
<tr>
<th>ID</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq1</td>
<td>{a, b}, {c}, {f}, {g}, {e}</td>
</tr>
<tr>
<td>seq2</td>
<td>{a, d}, {c}, {b}, {a, b, e, f}</td>
</tr>
<tr>
<td>seq3</td>
<td>{a}, {b}, {f}, {e}</td>
</tr>
<tr>
<td>seq4</td>
<td>{b}, {f, g}</td>
</tr>
</tbody>
</table>
Sequential Pattern Mining (SPM)

• SPM is probably the most popular set of techniques for discovering temporal patterns in sequence databases.

• SPM finds subsequences that are common to more than \textit{minsup} sequences.

• SPM is limited for making \textbf{predictions}. For example, consider the pattern \{x\},\{y\}. It is possible that \textit{y} appears frequently after an \textit{x} but that there are also many cases where \textit{x} is not followed by \textit{y}.

• For \textbf{prediction}, we need a measurement of the confidence that if \textit{x} occurs, \textit{y} will occur afterward.
Sequential Rule Mining (SRM)

• For prediction, an alternative is **Sequential Rule Mining**.

• A **sequential rule** typically has the form $X \rightarrow Y$ and has a **confidence** and a **support**.


• **SRM has several applications:** stock market analysis (Das et al., 1998; Hsieh et al., 2006), weather observation (Hamilton & Karimi, 2005), drought management (Harms et al. 2002), alarm analysis, etc.
Mining Sequential Rules in Sequence Databases

• A **sequential rule** \( X \Rightarrow Y \) is a relationship between two disjoint and non empty itemsets \( X,Y \).

• A sequential rule \( X \Rightarrow Y \) has **two properties**:
  – **Support**: the number of sequences where \( X \) occurs before \( Y \), divided by the number of sequences.
  – **Confidence** the number of sequences where \( X \) occurs before \( Y \), divided by the number of sequences where \( X \) occurs.

• **The task**: finding all **valid rules**, rules with a support and confidence not less than user-defined thresholds \( \minSup \) and \( \minConf \) (Fournier-Viger, 2010).
An example of Sequential Rule Mining

Consider $minSup = 0.5$ and $minConf = 0.5$:

A sequence database

<table>
<thead>
<tr>
<th>ID</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq1</td>
<td>{a, b}, {c}, {f}, {g}, {e}</td>
</tr>
<tr>
<td>seq2</td>
<td>{a, d}, {c}, {b}, {a, b, e, f}</td>
</tr>
<tr>
<td>seq3</td>
<td>{a}, {b}, {f}, {e}</td>
</tr>
<tr>
<td>seq4</td>
<td>{b}, {f, g}</td>
</tr>
</tbody>
</table>

Some rules found

<table>
<thead>
<tr>
<th>ID</th>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>{a, b, c} $\Rightarrow$ {e}</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>r2</td>
<td>{a} $\Rightarrow$ {c, e, f}</td>
<td>0.5</td>
<td>0.66</td>
</tr>
<tr>
<td>r3</td>
<td>{a, b} $\Rightarrow$ {e, f}</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>r4</td>
<td>{b} $\Rightarrow$ {e, f}</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>r5</td>
<td>{a} $\Rightarrow$ {e, f}</td>
<td>0.75</td>
<td>1.0</td>
</tr>
<tr>
<td>r6</td>
<td>{c} $\Rightarrow$ {f}</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>r7</td>
<td>{a} $\Rightarrow$ {b}</td>
<td>0.5</td>
<td>0.66</td>
</tr>
</tbody>
</table>

...
Current Algorithms

- **CMRules**: An association rule mining based algorithm for the discovery of sequential rules.
- **CMDeo**: An Apriori based algorithm for the discovery of sequential rules.
- **Limitation**: Both algorithms use a «generate-candidate-and-test» approach that may generate a large amount of candidates for dense datasets. Many candidates do not appear in the database.
RuleGrowth

• Main idea:

1. First, scan database to find frequent items. e.g. \{a, b, c, d \ldots\}
2. For each pairs of such items, try to create a rule with only two items. e.g. \{a\} ⇒ \{b\}.
3. Then, find larger rules by recursively scanning the database for adding a single item at a time to the left or right part of each rule (left and right expansions).
   e.g. \{a,c\} ⇒ \{b\}, then \{a,c,d\} ⇒ \{b\}, etc.
4. Each rule created is tested to see if it is valid.
RuleGrowth (2)

• When a rule should be expanded?
  **Property:** Adding an item to a rule results in a rule having a support that is lower or equal. Therefore, a rule should be expanded only if it has the minimum support.

• When a rule should be outputed?
  If the support and confidence and support are higher or equal to \textit{minsup} and \textit{minconf}. 
RuleGrowth (3)

• How to choose items for performing left expansions of a rule $X \Rightarrow Y$?
  \(\rightarrow\) Scan the sequences containing the rule and note items appearing in at least $\text{minsup}$ sequences before the last occurrence of $Y$.

• How to choose items for performing right expansions of a rule $X \Rightarrow Y$?
  \(\rightarrow\) Scan the sequences containing the rule and note items appearing in at least $\text{minsup}$ sequences after the first occurrence of $X$. 
RuleGrowth (4)

• How to avoid generating the same rules twice?
  → Add only an item to the left/right part of a rule if the item is larger than all items already in the left/right part.
  → Do not allow performing a left expansion after a right expansion. But allow performing a right expansion after a left expansion.
RuleGrowth (5)

Optimization: During the first database scan, record the first and last occurrence of each item for each sequence.

• This allows to create initial rules very efficiently.
• This allows to avoid scanning sequences completely when searching for items for expansions.

Optimization 2: The set of sequences containing X, Y, and X⇒Y is maintained for each rule X⇒Y so that the confidence can be calculated efficiently.
Performance Evaluation

- RuleGrowth, CMRules and CMDEO.
- Java, 1GB of RAM
- Three real-life public datasets.

<table>
<thead>
<tr>
<th></th>
<th>Kosarak</th>
<th>BMS</th>
<th>Toxin-Snake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence count</td>
<td>70,000</td>
<td>59,601</td>
<td>163</td>
</tr>
<tr>
<td>Item count</td>
<td>21,144</td>
<td>497</td>
<td>20</td>
</tr>
<tr>
<td>Average item count by sequence</td>
<td>7.97</td>
<td>2.51</td>
<td>60.61</td>
</tr>
<tr>
<td>Average different item count by sequence</td>
<td>7.97</td>
<td>2.51</td>
<td>17.84</td>
</tr>
</tbody>
</table>
Influence of $\text{minsup}$
Influence of $\text{minsup}$

Kosarak
Influence of \textit{minconf}

Kosarak

BMS

Snake
Conclusion

RuleGrowth,

• Is a novel algorithm for mining sequential rules common to several sequences,

• outperforms CMRules and CMDeo in terms of execution time and memory usage.

The Java source code can be downloaded as part of the Sequential Pattern Mining Framework at http://goo.gl/xat4k
Thank you. Questions?

Thanks to the NSERC and FQRNT funding programs.

Thanks to the organizers of the SAC 2011 Conference & DM track!
Application in a Cognitive Agent

- The **CTS cognitive agent** has predefined behaviors.
- Each execution of CTS is recorded as a sequence in a **sequence database**
- **Items** are actions and perceptions of CTS
- **Rules** are used for making predictions about which behavior will be successful with learners, and adapt the behavior of CTS accordingly.