Part 2

Mining Patterns in Sequential Data
Sequential Pattern Mining: Definition

“Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, sequential pattern mining is to find all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.”

~ [Agrawal & Srikant, 1995]

“Given a set of data sequences, the problem is to discover sub-sequences that are frequent, i.e., the percentage of data sequences containing them exceeds a user-specified minimum support.”

~ [Garofalakis, 1999]
Why Sequential Patterns?

Direct Knowledge

Feature Detection
Notation & Terminology

• Data:
  – Dataset: set of sequences
  – Sequence: an ordered list of itemsets (events) \(<e_1, \ldots, e_n>\)
  – Itemset: an (unordered) set of items \(e_i = \{i_{i1}, \ldots, i_{iz}\}\)

• \(S_{\text{sub}} = <s_1, \ldots, s_n>\) is a subsequence of sequence \(S_{\text{ref}} = <r_1, \ldots, r_n>\) if:
  \[\exists i_1 < \cdots < i_n: s_k \subseteq r_{i_k}\]

Example:

\[<a, (b,c), c>\quad \text{is subsequence}\]

\[<a, (d,e), (b,c), (a,c)>\]

• Length of a sequence: # items used in the sequence (not unique):
  Example: length \(<a,(b,c),a>\) = 4

More Examples: jupyter
Frequent Sequential Patterns

- Support $\text{sup}(S)$ of a (sub-)sequence $S$ in a dataset:
  Number of sequences in the dataset that have $S$ as a subsequence

- Given a user chosen constant $\text{minSupport}$:
  Sequence $S$ is **frequent** in a dataset if $\text{sup}(S) \geq \text{minSupport}$

- Task: Find all frequent sequences in the dataset

- If all sequences contain exactly one event:
  Frequent itemset mining!
Pattern Space

- General approach: enumerate candidates and count
- Problem: “combinatorial explosion”: Too many candidates
- Candidates for only 3 items:
  - Length 1: 100 candidates
  - Length 2: $100 \times 100 \times \frac{100 \times 99}{2} = 14,950$
  - Length 3: $\sum_{i=1}^{100} \#candidates\ for\ length\ i = 2^{100} - 1 \approx 10^{30}$

- Candidates for 100 items:
  - Length 1: 3 candidates
  - Length 2: 12 candidates
  - Length 3: 46 candidates
Monotonicity and Pruning

• If S is a subsequence of R \(\rightarrow\) then \(\text{sup}(S)\) is at most as large as \(\text{sup}(R)\)
• Monotonicity:
  If S is not frequent, then it is impossible that R is frequent!
  E.g. \(<a>\) occurs only 5 times, then \(<a, b>\) can occur at most 5 times
• Pruning:
  If we know that S is not frequent, we do not have to evaluate any supersequence of S!

Assume b is not frequent

Length 2: only 5 candidates

Length 3: only 20 candidates left
Apriori Algorithm (for Sequential Patterns) [Agrawal & Srikant, 1995]

• Evaluate pattern “levelwise” according to their length:
  – Find frequent patterns with length 1
  – Use these to find frequent patterns with length 2
  – ...

• First find frequent single items

• At each level do:
  – Generate candidates from frequent patterns of the last level
    • For each pair of candidate sequences (A, B):
      – Remove first item of A and the last item of B
      – If these are then equal:
        generate a new candidate by adding the last item of B at the end of A
    • E.g.: A = <a, (b,c), d>, B = <(b,c), (d,e)> \(\rightarrow\) new candidate <a, (b,c), (d,e)>

  – Prune the candidates (check if all subsequences are frequent)
  – Check the remaining candidates by counting

More Examples: jupyter
Extensions based on Apriori:

• Generalized Sequential Patterns (GSP): [Srikant & Agrawal 1996]
  – Adds max/min gaps,
  – Taxonomies for items,
  – Efficiency improvements through hashing structures

• PSP: [Masseglia et al. 1998]
  Organizes candidates in a prefix tree

• Maximal Sequential Patterns using Sampling (MSPS): Sampling
  [Luo & Choung 2005]

• ...

• See Mooney / Roddick for more details [Mooney & Roddick 2013]
SPaDE: Sequential Pattern Discovery using Equivalence Classes

- Uses a vertical data representation:

  (Original) Horizontal database layout

<table>
<thead>
<tr>
<th>SID</th>
<th>Time</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>a, b, d</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>b, d</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>b, c, d</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>b, d</td>
</tr>
</tbody>
</table>

  Vertical database layout

<table>
<thead>
<tr>
<th>SID</th>
<th>Time</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

- ID-lists for longer candidates are constructed from shorter candidates
- Exploits equivalence classes:
  - <b> and <d> are equivalent → <b, x> and <d, x> have the same support
- Can traverse search space with depth-first or breadth-first search
Extensions based on SPaDE

• **SPAM**: Bitset representation [Ayres et al. 2002]

• **LAPIN**: [Yang & et al. 2007]
  Uses last position of items in sequence to reduce generated candidates

• **LAPIN-SPAM**: combines both ideas [Yang & Kitsuregawa 2005]

• **IBM**: [Savary & Zeitouni 2005]
  Combines several datastructures (bitsets, indices, additional tables)
PrefixSpan [Pei et al. 2001]

- Similar idea to Frequent Pattern Growth in FIM
- Determine frequent single items (e.g., a, b, c, d, e):
  - First mine all frequent sequences starting with prefix <a…>
  - Then mine all frequent sequences starting with prefix <b…>
  - ...
- Mining all frequent sequences starting with <a…> does not require complete dataset!
- Build projected databases:
  - Use only sequences containing a
  - For each sequence containing a only use the part “after” a

<table>
<thead>
<tr>
<th>Given Sequence</th>
<th>Projection to a</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; b, (c,d), a, (b d), e &gt;</td>
<td>&lt;a, (b,d), e&gt;</td>
</tr>
<tr>
<td>&lt;c, (a,d), b, (d,e)&gt;</td>
<td>&lt;(a,d), b, (d,e)&gt;</td>
</tr>
<tr>
<td>&lt;b, (de), c&gt;</td>
<td>[will be removed]</td>
</tr>
</tbody>
</table>
PrefixSpan (continued)

- Given prefix a and projected database for a: mine recursively!
  - Mine frequent single items in projected database (e.g., b, c, d)
  - Mine frequent sequences with prefix <a, b>
  - Mine frequent sequences with prefix <a, c>
  - ...
  - Mine frequent sequences with prefix <(a,b)>
  - Mine frequent sequences with prefix <(a,c)>
  - ...
- Depth-First-Search
Advantages of PrefixSpan

• Advantages compared to Apriori:
  + No explicit candidate generation, no checking of not occurring candidates
  + Projected databases keep shrinking

• Disadvantage:
  − Construction of projected database can be costly
So... which algorithm should you use?

- All algorithms give the same result
- Runtime / memory usage varies

- Current studies are inconclusive
- Depends on dataset characteristics:
  - Dense data tends to favor SPaDE-like algorithms
  - Sparse data tends to favor PrefixSpan and variations

- Depends on implementations

P. Singer, F. Lemmerich: Analyzing Sequential User Behavior on the Web
The Redundancy Problem

- The result set often contains many and many similar sequences
- Example: find frequent sequences with minSupport = 10
  - Assume \(<a, (bc), d>\) is frequent
  - Then the following sequence also MUST be frequent:
    \(<a>, <b>, <c>, <a, b>, <a, c>, <a, d>, <b, d>, <c, d>, <(b,c)>, <a, (b,c)>, <a, b, d>, <a, c, d>, <(b,c), d>\)

- Presenting all these as frequent subsequences carries little additional information!
Closed and Maximal Patterns

- Idea: Do not use all patterns, but only...
  - ... frequent *closed* sequences:
    all super-sequences have a smaller support
  - ... frequent *maximal* sequences:
    All super-sequences are not frequent

- Example:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>sup (&lt;a, c&gt;) = 3 → frequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;a, b, c, d, e, f&gt;</td>
<td>sup (&lt;a, c, d&gt;) = 3 → frequent, closed</td>
</tr>
<tr>
<td>&lt;a, c, d&gt;</td>
<td>sup (&lt;a, c, d, e&gt;) = 2 → frequent, closed, max.</td>
</tr>
<tr>
<td>&lt;c, b, a&gt;</td>
<td>sup (&lt;a, c, d, e, f&gt;) = 1 → not frequent</td>
</tr>
<tr>
<td>&lt;b, a, (de)&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;b, a, c, d, e&gt;</td>
<td></td>
</tr>
</tbody>
</table>

- Set of all frequent sequences can be derived from the maximal sequences
- Count of all frequent sequences can be derived from the closed sequences
Mining Closed & Maximal Patterns

• In principle: can filter resulting frequent itemsets
• Specialized algorithms
  – Apply pruning during the search process
  – Much faster than mining all frequent sequences
• Some examples
  – Closed:
    • CloSpan: PrefixTree with additional pruning [Yan et al. 2003]
    • BIDE: Memory-efficient forward/backward checking [Wang&Han 2007]
    • ClaSP: Based on SPaDE [Gomariz et al. 2013]
  – Maximal:
    • AprioriAdjust: Based on Apriori [Lu & Li 2004]
    • VMSP: Based on vertical data structures [Fournier-Viger et al. 2014]
    • MaxSP: Inspired by PrefixSpan, maximal backward and forward extensions [Fournier-Viger et al. 2013]
    • MSPX: approximate algorithm using samples [Luo & Chung 2005]
Beyond Frequency

- Frequent sequence ≠ interesting sequence
- Example for text sequences:
  Most frequent sequences in “Adventures from Tom Sawyer”¹:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Support (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;and, and&gt;</td>
<td>13%</td>
</tr>
<tr>
<td>&lt;and, to&gt;</td>
<td>9.8%</td>
</tr>
<tr>
<td>&lt;to, and&gt;</td>
<td>9.1%</td>
</tr>
<tr>
<td>&lt;of, and&gt;</td>
<td>8.6%</td>
</tr>
</tbody>
</table>

- Two options:
  - Add constraints (filter)
  - Use interestingness measures

¹ According to [Petitjean et al. 2015]
Constraints

- **Item constraints**: e.g., high-utility items: Sum all items in the sequence > 1000$
- **Length constraint**: Minimum/maximum number of events/transactions
- **Model-based constraints**: Sub-/supersequences of a given sequence
- **Gap constraints**: Maximum gap between events of a sequence
- **Time constraints**: Given timestamps, maximum time between events of a sequence
- Closed or maximal sequences only
- ...
- Computation:

  1. Mine all frequent patterns
  2. Filter

  “Push constraints into the algorithm”
Interestingness Measures and top-k Search

• Use *interestingness measures*
  – Function that assign a numeric value (score) to each sequence
  – Should reflect the “assumed interestingness” for users
  – Desired properties:
    conciseness, generality, reliability, diversity, novelty, surprisingness, applicability

• New goal: search for the $k$ sequences that achieve the highest score

• Interestingness measure also implies a ranking of the result

• Simple mining approach:
  1. Compute all frequent patterns
  2. Compute the score of each pattern
**Confidence**

- Typical measure for association rule mining
- Can easily be adapted for sequential pattern
- Split sequence into a rule (e.g., with the last event as rule head)
- Confidence = accuracy of this rule

\[
\text{Confidence} (< A, B, D \Rightarrow F) = \frac{20}{30}
\]

- Can be used as a constraint or as an interestingness measure
Leverage [Petitjean et al. 2015]

• Compare support of a sequence with “expected support”:
  \[
  \text{Score}(S) = \text{sup}(S) - \text{expectedSupport}(S)
  \]

• Idea of expected support?

If:

- frequent

AND

- frequent

Then:

- frequent

Is also more likely to be frequent then the average 4-sequence.
It should be reported only, if its frequency exceeds expectation.

• Formalization for 2-sequences:

\[
\text{expectedSupport}(<a, b>) = \frac{\text{sup}(<a, b>) + \text{sup}(<b, a>)}{2}
\]

• Formalization for larger sequences generalizes this.
Other Interestingness Measures

- **Information theoretic approaches**: [Tatti & Vreeken 2012], [Lam et al. 2014]
  - Use minimum description length
  - Find sequence (sets) that best explain/compress the dataset

- **Model-based approaches** [Gwadera & Crestani 2010], [Lam et al. 2014]
  - Build a reference model (e.g., learn a markov chain model)
  - Determine which sequences are most unlikely given that model
  - (Compute statistical significance)

- Include time information
- ...

P. Singer, F. Lemmerich: Analyzing Sequential User Behavior on the Web
Efficient top-k Sequential Pattern Mining [Petitjean et al. 2015]

- Example Algorithm SkOPUS:

- Depth First Search
- Pruning:
  - Interestingness measures like leverage/confidence are not directly monotone (unlike support)
    E.g.: score (⟨a, b, c⟩) can be higher then score (⟨a, b⟩)
  - Use upper bounds (“optimistic estimates”) oe(S)
    For each sequence S this is threshold, such that no super-sequence of S has a higher score
  - Has to be determined for each interestingness measure separately
  - Often easy to compute for a single interestingness measure
Case Study Web Log Mining [Soares et al. 2006]

• Portuguese web portal for business executives:
  • Data: 3,000 users; 70,000 session; 1.7M accesses

• Navigation patterns found on page level:
  – Too many
  – Not very useful

• On type level ("news", "navigation")
  – More interesting findings
Mining Web logs to Improve Website Organization

- Given link structure of a web page, visitor log
- Build sequences for each visitor
- Define target page

- Find frequent paths to the target page
- Identify links that could shorten user paths

[Srikant & Yang 2001]
Available Software Libraries

• **Java:**
  – SPMF (most extensive library)
  – Basic support in RapidMiner, KNIME

• **R**
  – arulesSequences package
  – TraMiner package

• **Python**
  – Multiple basic implementations
  – The implementations for this tutorial (mainly educational, not efficient)

• **Spark:** PrefixSpan available
What we did not talk about...

• Episode mining  [Mannila et al. 1997]
  – Given long sequences: find recurring patterns
  – Mining: candidate generation vs. pattern growth

• Discriminative sequential pattern

• Incremental mining / data streams

• Pattern in time-series
REFERENCES (1/2)


icons in this slide set are CC0 Public Domain, taken from pixabay.com
References (2/2)


