The Long and the Short of It

summarising event sequences with serial episodes

Nikolaj Tatti & Jilles Vreeken
Question of the day

How can we discover the key patterns from an event sequence?
Summarising Event Sequences

The **ideal** outcome of pattern mining
- patterns that show the structure of the data
- preferably a small set, without redundancy or noise

Frequent pattern mining does **not** achieve this
- pattern explosion → overly many, overly redundant results

We pursue the ideal for serial episodes
- we want a group of patterns that summarise the data well
- we take a **pattern set** mining approach
Event sequences

Alphabet $\Omega$  
\{ $a, b, c, d, \ldots$ \}

Data $D$  
one, or multiple sequences

\{ $a b d c a d b a a b c, a b d c a d b, a b d c a d b a a, \ldots$ \}
Event sequences

Alphabet $\Omega$ 

\{ $a$, $b$, $c$, $d$, ... \}

Data $D$

one, or multiple sequences

{ $a$, $b$, $d$, $c$, $a$, $d$, $b$, $a$, $a$, $b$, $c$, $a$, $d$, $a$, $b$, $a$, $b$, $c$ }

Patterns

serial episodes

‘subsequences allowing gaps’
Event sequences

Alphabet $\Omega$ \{ $a$, $b$, $c$, $d$, ... \}

Data $D$

one, or multiple sequences

Patterns

serial episodes

'subsequences allowing gaps'
Summarising Event Sequences

We want to find good summaries.

Three important questions
1. how do we score a pattern-based summary?
2. how do we describe a sequence given a pattern set?
3. how do we find good pattern sets?
Summarising Event Sequences

We want to find good summaries.

Three important questions
1. how do we score a pattern-based summary?
2. how do we describe a sequence given a pattern set?
3. how do we find good pattern sets?
Scoring a Summary

We want models that generalise the data and hence, we need a score that
- **rewards** models that identify real structure, and
- **punishes** redundancy and noise

No off-the-shelf score available for serial episodes
- e.g. no well-founded priors
- we can, however, make these goals concrete by **MDL**
The Minimum Description Length (MDL) principle

given a set of models \( \mathcal{M} \), the best model \( M \in \mathcal{M} \) is that \( M \) that minimises

\[
L(M) + L(D|M)
\]

in which

\( L(M) \) is the length, in bits, of the description of \( M \)

\( L(D|M) \) is the length, in bits, of the description of the data when encoded using \( M \)

(see, e.g., Rissanen 1978, 1983, Grünwald, 2007)
MDL for Event Sequences

By MDL we define

the optimal set of serial episodes as the set
that describes the data most succinctly

To use MDL, we need

- a lossless encoding for our models,
- a lossless encoding for the data given a model

(for itemsets, see Siebes et al, 2006, Vreeken et al 2011)
Models

As models we use **code tables**
- dictionaries of patterns and associated codes

<table>
<thead>
<tr>
<th>pattern</th>
<th>code</th>
<th>gap</th>
<th>non-gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>p</td>
<td>?</td>
<td>!</td>
</tr>
<tr>
<td>da</td>
<td>q</td>
<td>?</td>
<td>!</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Encoding Event Sequences

Data $D$: $a b d c a d b a a b c$

Encoding 1: using only singletons

$C_p = [a b d c a d b a a b c]$

$CT_1 = [a a b b c c d d]$

The length of the code $X$ for pattern $X$

$$L(X) = - \log(p(X)) = - \log\left(\frac{\text{usg}(X)}{\sum \text{usg}(Y)}\right)$$

The length of the code stream

$$L(C_p) = \sum_{X \in CT} \text{usg}(X) L(X)$$
Encoding Event Sequences

Data $D$: $a b d c a d b a a b c$

Encoding 2: using patterns

$C_p$: $p d a q b p$

$C_g$: $! ? ! ? ! ! !$

Alignment: $a b d c a d b a a b c$

$CT_2$: $a a b b c c d d p ? ! q ? !$

$gap$

$gap$
Encoding Event Sequences

Data $D$: $a\ b\ d\ c\ a\ d\ b\ a\ a\ b\ c$

Encoding 2: using patterns

$C_p$: $p\ d\ a\ q\ b\ p$

$C_g$: $!\ ?\ !\ ?\ !\ !\ !$

The length of a gap code $\square$ for pattern $X$

$$L(\square) = -\log(p(\square | p))$$

and analogue for non-gap codes $!$
Encoding Event Sequences

By which, the encoded size of $D$ given $CT$ and $C$ is

$$L(D \mid CT) = L(C_p \mid CT) + L(C_g \mid CT)$$

...skipping the details of $L(CT \mid C)$...

Then, our goal is to minimise

$$L(CT, D) = L(CT \mid C) + L(D \mid CT)$$
Summarising Event Sequences

We want to find good summaries.

Three important questions
1. how do we score a summary?
2. how do we describe a sequence given a pattern set?
3. how do we find good pattern sets?
How to Cover your String

There are many ways $C$ to describe a sequence given a set of patterns. We are after the **optimum**.

$$a\ b\ d\ c\ a\ d\ b\ a\ a\ b\ c$$

$$p\ d\ a\ q\ b\ p$$

$$!\ ?\ ?\ !\ ?\ !\ !\ !\ !$$

$$p\ d\ c\ a\ d\ b\ a\ a\ b$$

$$!\ ?\ ?\ ?\ ?\ ?\ ?\ ?\ ?\ ?\ ?\ !$$

$$a\ b\ q\ c\ q\ b\ p$$

$$?\ !\ ?\ !\ ?\ !\ !\ !$$

$$CT: \ a\ b\ b\ d\ d\ c\ c\ abc\ p\ ?\ !$$

$$da\ q\ ?\ ?\ !$$

etc...
How to Cover your String

There are many ways $C$ to describe a sequence given a set of patterns. We are after the optimum.

1. if we fix the cover, we can obtain the optimal code lengths
2. if we fix the code lengths, we can obtain the optimal cover by dynamic programming

We alternate these steps until convergence
How to Cover your String

There are many ways to describe a sequence given a set of patterns. We are after the optimum.

1. if we fix the cover, we can obtain the optimal code lengths
2. if we fix the code lengths, we can obtain the optimal cover by dynamic programming

We alternate these steps until convergence.
How to Cover your String

There are many ways to describe a sequence given a set of patterns. We are after the optimum.

1. If we fix the cover, we can obtain the optimal code lengths.
2. If we fix the code lengths, we can obtain the optimal cover by dynamic programming.

We alternate these steps until convergence.

SQS

Summarising event seQuenceS
Summarising Event Sequences

We want to find good summaries.

Three important questions
1. how do we score a summary?
2. how do we describe a sequence given a pattern set?
3. how do we find good pattern sets?
Mining Code Tables

There are very many possible pattern sets. We are after the optimum

However, the search space is huge, complex, and does not exhibit trivial structure

We propose two algorithms for mining code tables

- **SQS-CANDS** filters ordered lists of pre-mined candidates
- **SQS-SEARCH** mines good code tables directly from data
**SQS-CANDIDATES**

- Database
- Many many patterns
- SQS-CANDS: select pattern, accept/reject, add to code table, compress database
- MDL
- Code table
SQS-SEARCH

SQS-SEARCH

compress database

MDL

accept/reject

generate candidates

add to code table

select pattern

Database

Code table
Experiments

- **synthetic data**
  - random
  - HMM
    - no structure found

- **real data**
  - text data
    - structure recovered for interpretation

|               | |Ω| |D| |F| |P| |P| |ΔL|
|---------------|--------|------|-----|-----|-----|-----|-----|-----|-----|
| **Addresses** | 5 295  | 56   | 15 506 | 138 | 155 | 5k  |
| **JMLR**      | 3 846  | 788  | 40 879 | 563 | 580 | 30k |
| **Moby Dick** | 10 277 | 1    | 22 559 | 215 | 231 | 10k |

(implementation available at http://adrem.ua.ac.be/sqs)
SQS-CANDIDATES

Compression improves with richer candidate sets i.e. lower support thresholds

![Graph showing compression improvement with support thresholds](image-url)
Optimising our Score

**Both strategies** show good convergence. **SQS-SEARCH** dips due to batch-wise search.
Selected Results

**JMLR**
- support vector machine
- machine learning
- state [of the] art
- data set
- Bayesian network

**Pres. Addresses**
- unit[ed] state[s]
- public econ. expenditur
- take oath
- equal right
- exercis power

(for SQS-SEARCH)
Conclusions

Mining informative sets of patterns
- is an important aspect of exploratory data mining

SQS approximates the ideal for serial episodes
- SQS-CANDS filters a pre-mined candidate list
- SQS-SEARCH mines good code tables directly from data

Future work includes
- richer data and pattern types
- applying SQS in real-world settings

(implementation available at http://adrem.ua.ac.be/sqs)
Thank you!

**Mining informative** sets of patterns
- is an important aspect of exploratory data mining

**SQS** approximates the ideal for serial episodes
- **SQS-CANDS** filters a pre-mined candidate list
- **SQS-SEARCH** mines good code tables directly from data

**Future work** includes
- richer data and pattern types
- applying SQS in real-world settings

(implementation available at [http://adrem.ua.ac.be/sqs](http://adrem.ua.ac.be/sqs))