Mining Sequential Patterns

Jilles Vreeken
How can we discover the key patterns from an event sequence?
abc, da + noise

(Tatti & Vreeken, KDD 2012)
First things first
What’s my signature?

data analysis ↔ communication

transfer the data to the analyst in as few as possible bits

‘induction by compression’
What does that mean?

defining well-founded objective functions for **exploratory** tasks

using **information theory**

for measuring how many bits of information a result gives

MDL, Kolmogorov Complexity, Kullback-Leibler, Maximum Entropy, (cumulative) entropy
and now to business...
Event sequences

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

discrete events,
e.g., words, alarms, etc.

Data $D$  
\[
\begin{array}{cccccccccccc}
  & a & b & d & c & a & d & b & a & a & b & c \\
\end{array}
\]
once, or multiple sequences

\[
\{ a b d c a d b a a b c , \\
a b d c a d b , \\
a b d c a d b a a , ... \}
\]
Event sequences

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

discrete events, e.g., words, alarms, etc.

Data $D$

one, or multiple sequences

Pattern Language

serial episodes

subsequences allowing for gaps
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
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
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
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**
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, 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 Vreeken et al 2011)
Models

As models we use **code tables**
- dictionaries of patterns & codes
- always contains all singletons

We use optimal prefix codes
- easy to compute,
- behave predictably,
- good results

<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 $X$

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

The length of the code stream

\[ L(C_p) = \sum_{X \in CT} usg(X)L(X) \]
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: $a$ $b$ $d$ $c$ $a$ $d$ $b$ $a$ $a$ $b$ $c$

$CT_2$: $a$ $a$

$gaps$ $non-gaps$

$abc$ $p$ $?$ $!

da$ $q$ $?$ !
Encoding Event Sequences

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

Encoding 2: using patterns

$C_p$ \[
\begin{array}{ccccc}
p & d & a & q & b & p \\
\end{array}
\]

$C_g$ \[
\begin{array}{cccccc}
\end{array}
\]

$CT_2$: \[
\begin{array}{c}
a \\
b \\
c \\
d \\
abc \\
da \\
a \\
\end{array}
\]

The length of a gap code $?$ for pattern $X$

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

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 valid C’s that describe a sequence given a set of patterns. We are after the optimum.

$$\begin{array}{ccccccc}
\end{array}$$

CT: $$\begin{array}{ccccccc}
a & b & b \\ b & b \\ c & c \\ abc & p & ? & ! \\
da & q & ? & ! \\
\end{array}$$

or, 

$$\begin{array}{ccccccc}
\end{array}$$

or, 

$$\begin{array}{ccccccc}
\end{array}$$

or, 

$$\begin{array}{ccccccc}
\end{array}$$

etc...
How to Cover your String

There are many valid C’s that describe a sequence given a set of patterns. We are after a **good** one.

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**
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

SQS-CANDS

select pattern

accept/reject

add to code table

MDL

compress database

Database

Many many patterns

Code table
SQS-SEARCH

Database → SQS-SEARCH → MDL
compress database → accept/reject
add to code table → generate candidates
select pattern → Code table
The Basic Idea

Given a code table and cover, how can we refine it?
- by checking if there are **patterns** in how the codes are used

Patterns in the code stream imply **unmodeled structure**!

\[ C_p \mid CT_0 : \quad a \quad b \quad c \quad d \quad a \quad d \quad b \quad a \quad a \quad b \quad c \quad d \quad \cdots \]

\[ a \rightarrow b \quad \text{happens a lot, let’s add it to } CT \]
Given a code table, how can we refine it?
- by checking if there are patterns in how the codes are used

Patterns in the code stream imply unmodeled structure

\[ C_p \mid CT_0 : \quad \text{a b c d a d b a a b c d} \quad \ldots \]

\[ C_p \mid CT_1 : \quad \text{p c d p d a p c d} \quad \ldots \quad p : \quad \text{a} \rightarrow \text{b} \]
The Basic Idea

Given a code table, how can we refine it?
- by checking if there are patterns in how the codes are used

Patterns in the code stream imply unmodeled structure

\[ C_p \mid CT_0 : \quad a \ b \ c \ d \ a \ d \ b \ a \ a \ b \ c \ d \ \cdots \]
\[ C_p \mid CT_1 : \quad p \ c \ d \ p \ d \ a \ p \ c \ d \ \cdots \quad p : \quad a \rightarrow b \]
\[ C_p \mid CT_2 : \quad p \ q \ p \ d \ a \ p \ q \ \cdots \quad q : \quad c \rightarrow d \]
The Basic Idea

Given a code table, how can we refine it?

- by checking if there are patterns in how the codes are used

Patterns in the code stream imply unmodeled structure

Given a code stream, generate all code pairs

- consider these as candidates, in order of estimated gain
  - when total encoded size decreases, re-generate and re-rank
The Basic Idea

Given a code table, how can we refine it?
- by checking if there are **patterns** in how the codes are used

Patterns in the code stream imply **unmodeled structure**

Given a code stream, generate all code pairs
- consider these as candidates, in order of estimated gain
- when batch is empty, re-generate and re-rank
Both strategies show good convergence. **SQS-Search** dips due to batch-wise search.
## Experiments

- **synthetic data**
  - random
  - HMM
  - ✓ no structure found
  - ✓ structure recovered
text for interpretation

- **real data**
  - various

|                | $|\Omega|$ | $|D|$ | $\#\text{freq ep.}$ | $|P|$ | $\Delta L$ |
|----------------|-----------|------|---------------------|------|-----------|
| Pres. Addresses| 5 295     | 62 066| 15 506              | 155  |
| JMLR           | 3 846     | 75 646| 40 879              | 580  |
| Moby Dick      | 10 277    | 105 719| 22 559             | 231  |

**Sqs-Search**
Results of SQs

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

**PRES. ADDRESSES**
- unit[ed] state[s]
- take oath
- army navy
- under circumst.
- econ. public expenditur

(top-5 from 563) (selection from top-25)
That was back in 2012

now back to 2015
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
Though nice, SQS is limited.
With SQUEEZE we aim to push the envelope.

1) richer pattern language
   serial episodes

\[ a \rightarrow b \rightarrow c \]
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
   serial episodes
   parallel episodes

\[
\begin{align*}
\text{a} \rightarrow \text{b} \rightarrow \text{c} \\
\text{a} \rightarrow \text{b} \rightarrow \text{d} \rightarrow \text{c} \\
\text{a} \rightarrow \text{d} \rightarrow \text{b} \rightarrow \text{c}
\end{align*}
\]
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language

serial episodes

parallel episodes
SQUEEZE

Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
   serial episodes
     \[ a ightarrow b ightarrow c \]
   parallel episodes
     \[ a ightarrow b \text{ and } d \rightarrow c \]
   ‘choice‘ episodes
     \[ a ightarrow d \rightarrow c \]
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
   - serial episodes
     a → b → c
   - parallel episodes
     a → b and d → c
   - ‘choice’ episodes
     a → b or d → c
**SQUEEZE**

Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language

- **serial episodes**
  - `a → b → c`

- **parallel episodes**
  - `a → b and d → c`

- **‘choice’ episodes**
  - `a → b or d → c`

- **‘stopisodes’**
Though nice, SQS is limited
With **SQUEEZE** we aim to push the envelope.

1) richer pattern language
   - serial episodes
   - parallel episodes
   - ‘choice’ episodes
   - ‘stopisodes’
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
   serial episodes
   parallel episodes
   ‘choice’ episodes
   ‘stopisodes’
Though nice, SQS is quite limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
2) better covers

\[ a \ b \ c \ d \ a \ d \ b \ a \ a \ b \ c \ a \ d \ a \ b \ a \ b \ c \]
Though nice, SQS is quite limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
2) better covers

SQS: non-overlapping, non-nested, non-interleaving
Though nice, SQS is quite limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
2) better covers

SQUEEZE: non-overlapping, non-nesting, non-interleaving
Though nice, SQS is quite limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language
2) better covers

SQUEEZE: non-overlapping, non-nesting, non-interleaving

(work in progress, with Bhattacharyya)
Though nice, SQS is quite limited
With Ditto we push the envelope to **multivariate** data & patterns

<table>
<thead>
<tr>
<th>Categorical</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^0$: a b c a ... b c a b a a</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>$S^1$: d e f d ... d f e f e d</td>
<td>d e</td>
<td>f d</td>
</tr>
<tr>
<td>$S^2$: g h i g ... g i h h i g</td>
<td>h i</td>
<td>g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^0$: a a a a a ... a a a a a a a</td>
<td>a a a a a</td>
</tr>
<tr>
<td>$S^1$: b b ... b b b b b</td>
<td>b</td>
</tr>
<tr>
<td>$S^2$: c c c ... c c c c c</td>
<td>c</td>
</tr>
</tbody>
</table>

(Bertens, Vreeken & Siebes, under submission)
We ran DITTO on translations of the same EU document, stemming, and removing stop words, aligning per sentence. For a minimal support of 10, among the top-ranked results,

Pattern 1:
- French: relev
- German: stellt fest dass
- English: note

Pattern 3:
- German: million eur
- English: eur million

Pattern 7:
- French: parl
- German: parlament
- English: parliament

\[ t_1 \quad t_2 \quad t_3 \quad t_4 \]
So, patterns, that is all?

No.

MDL scores can be seen as a **likelihood** score
- and... with such a score we can do all sorts of cool things

What I’ve been doing before
- classification
- missing value estimation
- clustering
- ...etc...

What I’m currently exploring
- measure `structuredness`
- noise reduction
- budgeted description
Conclusions

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

**SQs** approximates the ideal for serial episodes
- complex problem, fast heuristics
- **SQs** extracts good models directly from data

**Ongoing work** includes
- more complex data and pattern types
- applying **SQs** and friends in real-world settings

(implementations available)
Thank you!

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

**SQs** approximates the ideal for serial episodes
- **complex** problem, **fast** heuristics
- **SQs** extracts **good** models directly from data

**Ongoing work** includes
- more complex data and pattern types
- applying **SQs** and friends in real-world settings

(Implementations available)