The Long and the Short of It
summarising event sequences with serial episodes

Jilles Vreeken & Nikolaj Tatti
Question at hand

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

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

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

Data $D$
- One, or multiple sequences

Patterns
- Serial episodes

'Subsequences allowing 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

Frequent pattern mining does **not** achieve this
- pattern explosion $\rightarrow$ 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 summary?
2. how do we cover 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 for Event Sequences

By MDL we define

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

We use 2-part MDL

- we are interested in the patterns, after all

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 & codes
- always contains all singletons

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

<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

CT$_1$: a a b b c c d d

Encoding 1: using only singletons

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

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

$abc$ $da$ $q$$?$$!$

gap  gap  gap

gaps  non-gaps
Encoding Event Sequences

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

Encoding 2: using patterns

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

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

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

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

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

Next, we discuss how to compute $L(CT \mid C, D)$
Encoding a Code Table

$L(CT \mid C, D)$ consists of

X  X  ?  !  
Y  Y  ?  !  
a  a  
z  z  
Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in $D$

$$L_N(|\Omega|) + L_N(||D||) + \log\left(\frac{||D|| - 1}{|\Omega| - 1}\right)$$
Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in $D$

$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right)$$
Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in $D$

$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right)$$

2) number of patterns, total, and per pattern usage

$$L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$
Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in $D$

$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right)$$

2) number of patterns, total, and per pattern usage

$$L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$
Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in $D$
$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right)$$

2) number of patterns, total, and per pattern usage
$$L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$

3) per pattern $X$ : its length, elements, and number of gaps
$$L_N(|X|) - \left[ \sum_{x \in X} \log p(x \mid D) \right] + L_N(gaps(X) + 1)$$
Encoding a Code Table

\( L(CT \mid C, D) \) consists of

1) base singleton counts in \( D \)
\[
L_N(|\Omega|) + L_N(\|D\|) + \log \left( \frac{\|D\| - 1}{|\Omega| - 1} \right)
\]

2) number of patterns, total, and per pattern usage
\[
L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)
\]

3) per pattern \( X \): its length, elements, and number of gaps
\[
L_N(|X|) - \left[ \sum_{x \in X} \log p(x \mid D) \right] + L_N(gaps(X) + 1)
\]
Encoding Event Sequences

Now, our goal is to minimise

\[ L(CT, D) = L(CT | C, D) + L(D | CT) \]
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.

\[ a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c \]

or,

\[ p \ d \ a \ q \ b \ p \]

or,

\[ p \ d \ c \ a \ d \ b \ a \ a \ b \]

or,

\[ a \ b \ q \ c \ q \ b \ p \]

etc...
How to Cover your String

There are many ways $C$ to 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. Again, we are after the optimum

However, again, 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

MDL

compress database

add to code table

Code table
SQS-SEARCH

1. Database
2. Compress database
3. MDL
4. Accept/reject
5. Generate candidates
6. Select pattern
7. Add to code table
8. Code table
Experiments

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

|               | $|\Omega|$ | $|D|$ | # Cnds | $|P|$ | $|P|$ | $\Delta L$ |
|---------------|--------|------|--------|------|------|-----------|
| **Addresses** | 5 295  | 56   | 15 506 | 138  |      |           |
| **JMLR**      | 3 846  | 788  | 40 879 | 563  |      | 580       |
| **Moby Dick** | 10 277 | 1    | 22 559 | 215  |      | 231       |

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

Minining 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))