Mining Sequential Patterns

Jilles Vreeken

UNIVERSITÄT DES SAARLANDES

M2CI CLUSTER OF EXCELLENCE

max planck institut informatik

Question of the Day

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

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

$\boxed{abc}$, $\boxed{da}$ $+$ *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 Ω $\{ a, b, c, d, \ldots \}$

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

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

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 Ω $\{ a, b, c, d, \ldots \}$

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

Data D

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

one, or multiple sequences

$\{ a \; b \; d \; c \; a \; d \; b \; a \; a \; b \; c,$
$a \; b \; d \; c \; a \; d \; b,$
$a \; b \; d \; c \; a \; d \; b \; a \; a, \ldots \}$

Pattern Language

serial episodes

$a \rightarrow b$

$d \rightarrow b \rightarrow c$

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

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

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

pattern	code	gap	non-gap
abc	*p*	?	!
da	*q*	?	!
a	*a*	-	-
b	*b*	-	-
c	*c*	-	-
d	*d*	-	-

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

Encoding Event Sequences

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

Encoding 1: using only singletons

C_p $\boxed{a}\boxed{b}\boxed{d}\boxed{c}\boxed{a}\boxed{d}\boxed{b}\boxed{a}\boxed{a}\boxed{b}\boxed{c}$

CT_1:

a	$\boxed{a}$
b	$\boxed{b}$
c	$\boxed{c}$
d	$\boxed{d}$

The length of the code $\boxed{X}$ for pattern X

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

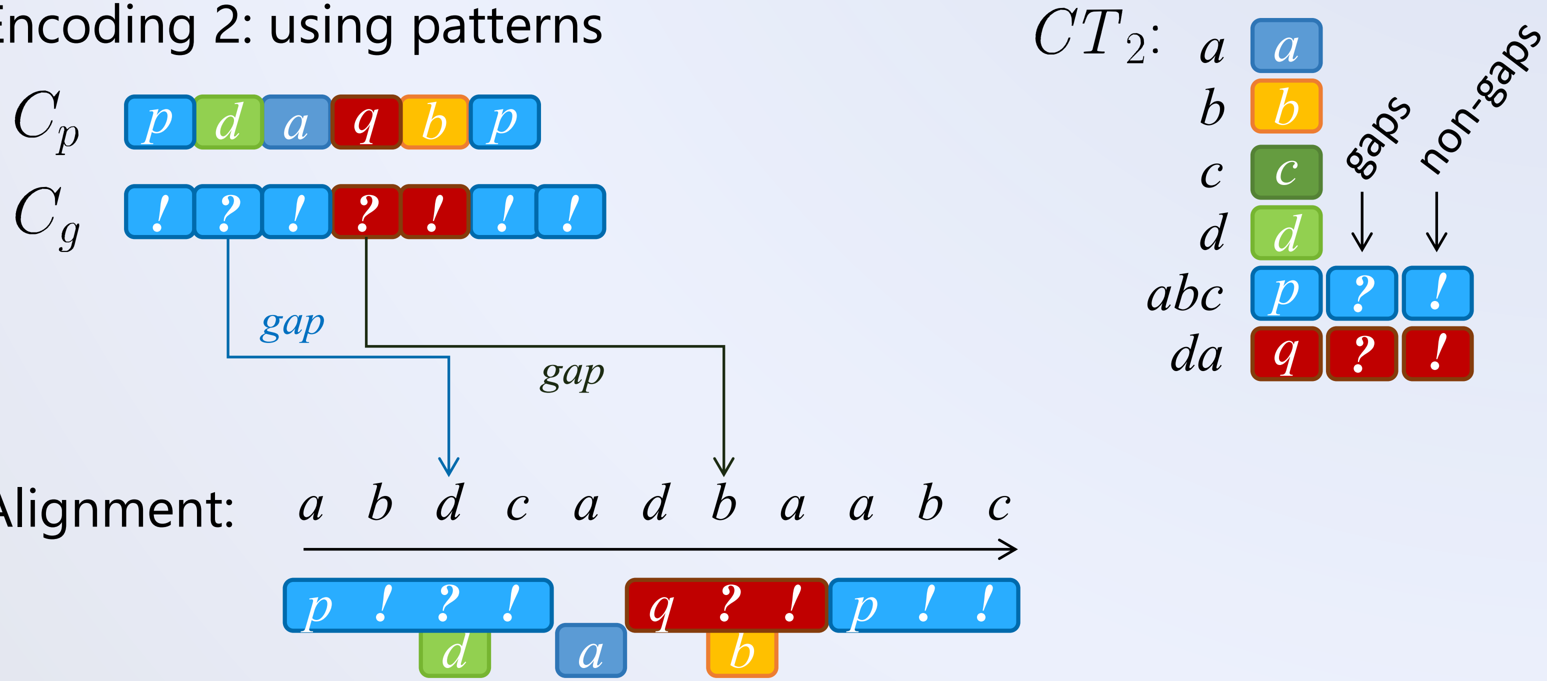
The length of the code stream

$$L(C_p) = \sum_{X \in CT} usg(X) L(\boxed{X})$$

Encoding Event Sequences

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

Encoding 2: using patterns



Encoding Event Sequences

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

Encoding 2: using patterns

C_p $\boxed{p}\boxed{d}\boxed{a}\boxed{q}\boxed{b}\boxed{p}$

C_g $\boxed{!}\boxed{?}\boxed{!}\boxed{?}\boxed{!}\boxed{!}\boxed{!}$

CT_2:

		gaps	non-gaps
a	$\boxed{a}$		
b	$\boxed{b}$		
c	$\boxed{c}$		
d	$\boxed{d}$		
abc	$\boxed{p}$	$\boxed{?}$	$\boxed{!}$
da	$\boxed{q}$	$\boxed{?}$	$\boxed{!}$

The length of a gap code $\boxed{?}$ for pattern X

$$L(\boxed{?}) = -\log(p(\boxed{?} \mid \boxed{p}))$$

and analogue for non-gap codes $\boxed{!}$

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

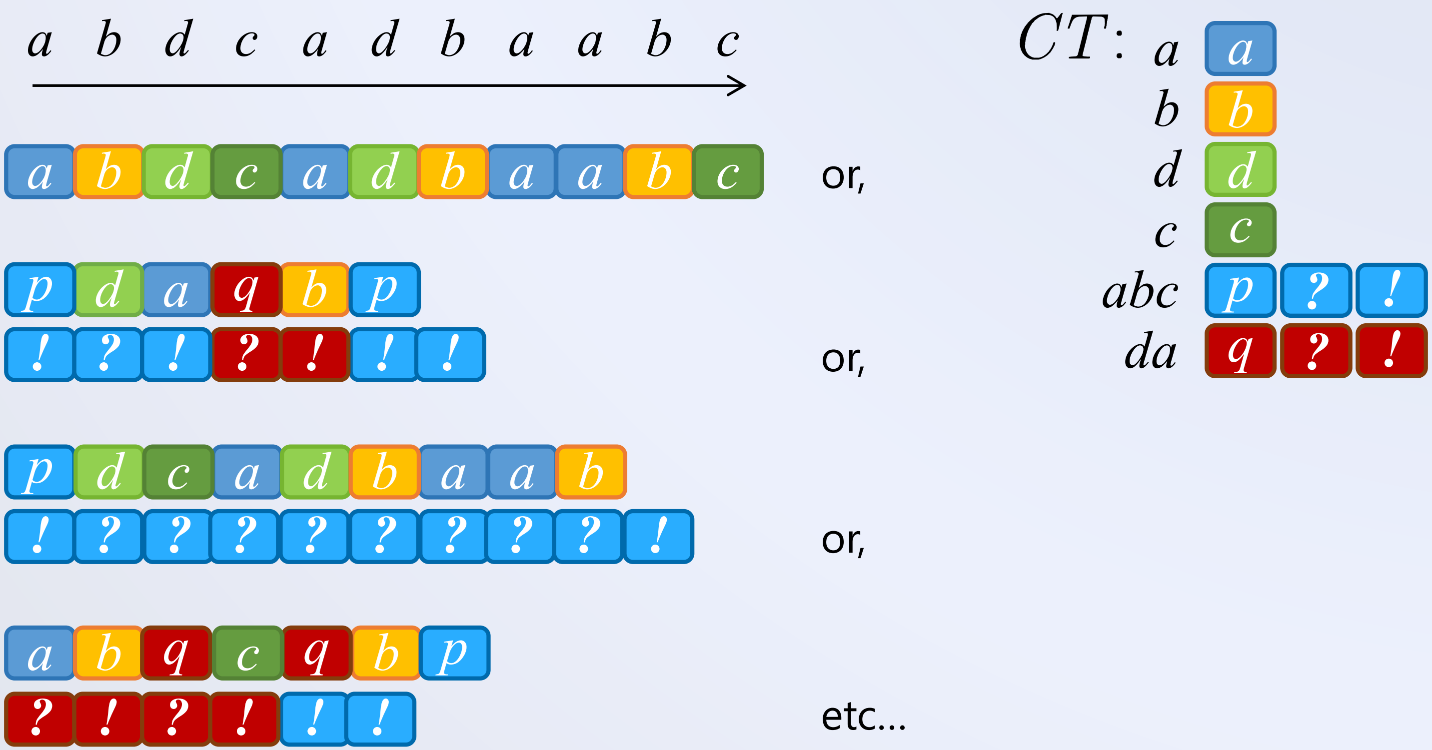
We want to find good summaries.

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.**



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

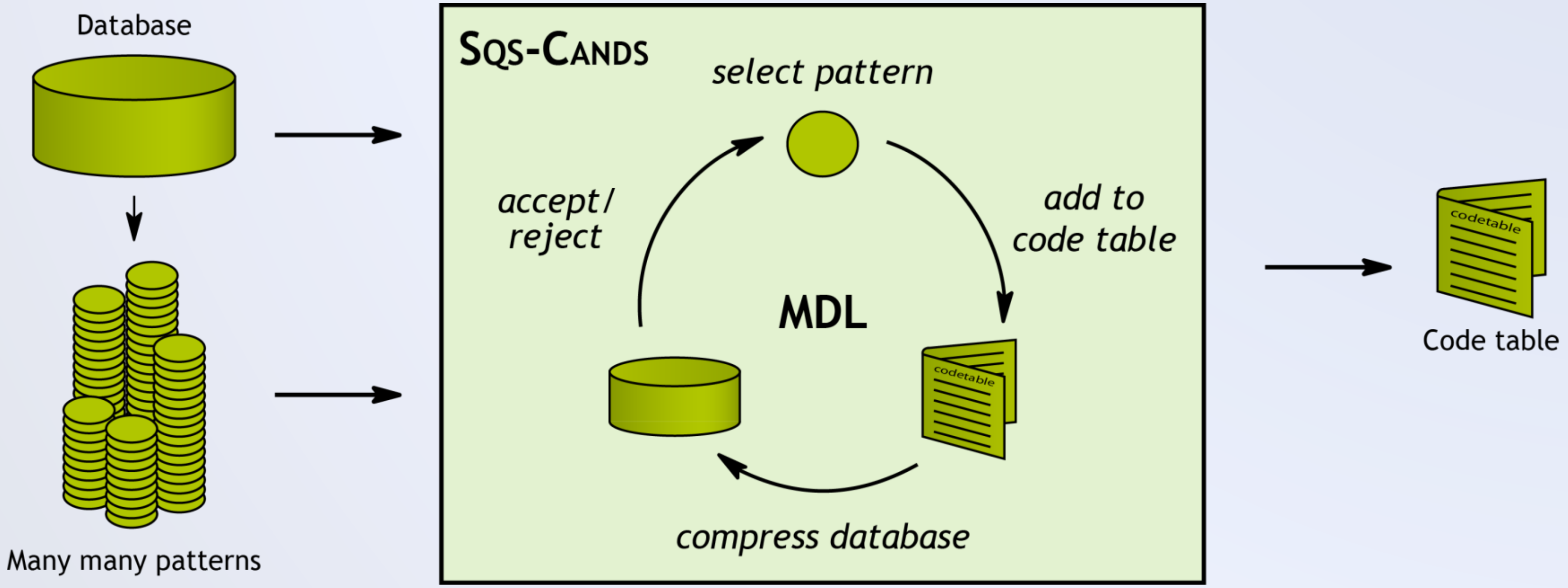
There are very many possible pattern sets.
We are after the **optimum**

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

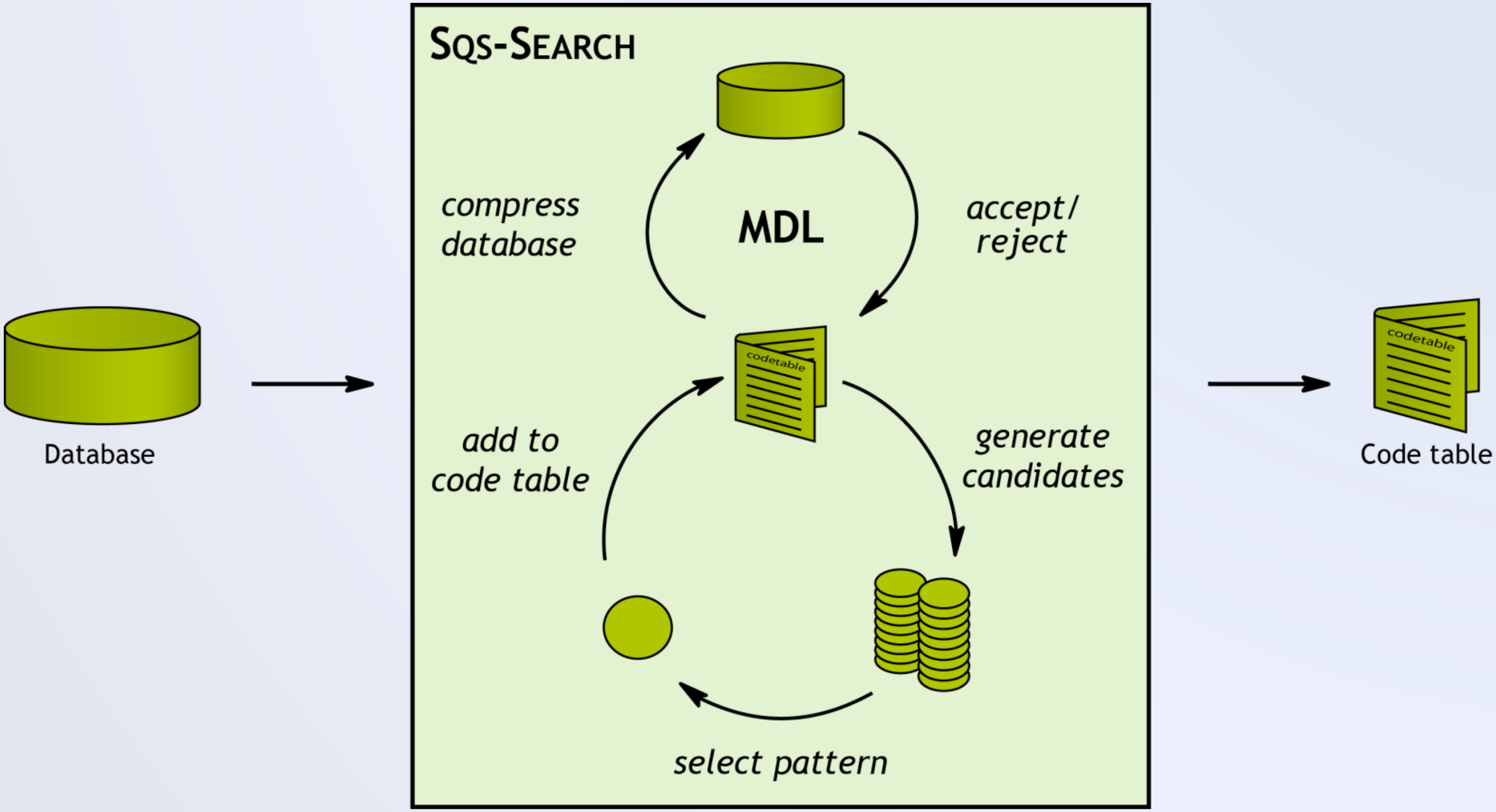
We propose two algorithms for mining code tables

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

Sqs-Candidates



Sqs-Search


Database
Sqs-Search
compress
database
MDL
accept/
reject
codetable
add to
code table
generate
candidates
select pattern
codetable
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!**

$C_p \mid CT_0$: $a\ b\ c\ d\ a\ d\ b\ a\ a\ b\ c\ d\ \cdots$

$a \rightarrow b$ happens a lot, let's add it to CT

The Basic Idea

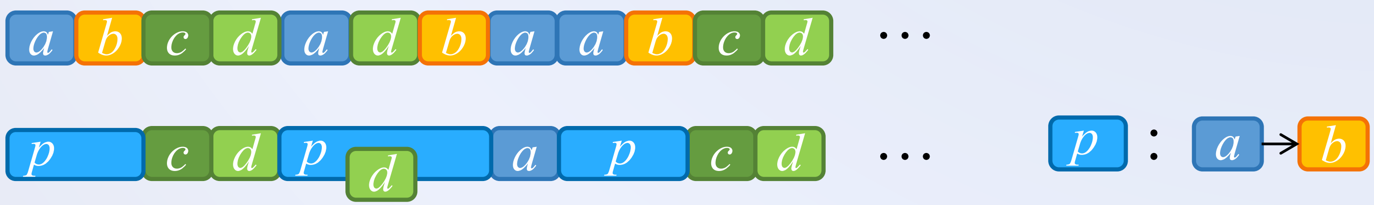
Given a code table, how can we refine it?

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

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

$C_p \mid CT_0 :$

$C_p \mid CT_1 :$

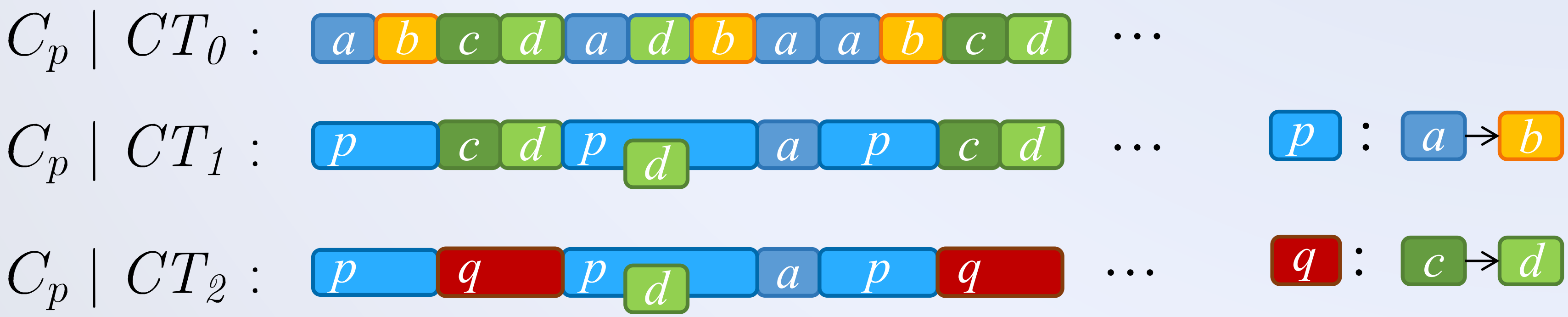


The Basic Idea

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**



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**

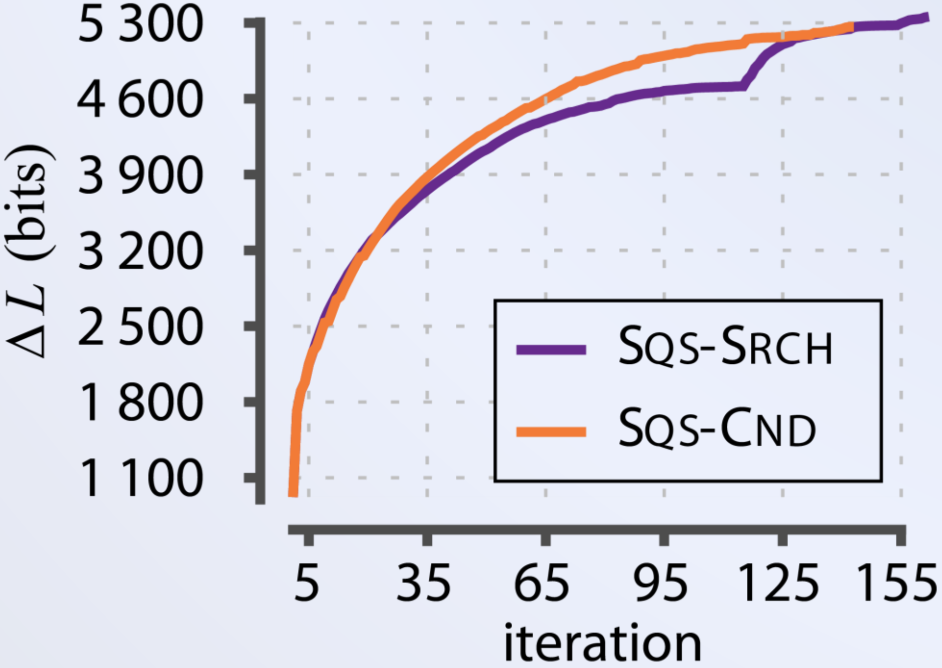
Given a code stream, generate all code pairs

- consider these as candidates, in order of estimated gain
- when batch is empty, re-generate and re-rank

Optimising our Score

Both strategies show good convergence

SQS-SEARCH dips due to batch-wise search



Experiments

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

	$\lvert\Omega\rvert$	$\lVert D\rVert$	$\#\,freq\ ep.$	**SQS-SEARCH** $\lvert\mathcal{P}\rvert$	ΔL
Pres. Addresses	5 295	62 066	15 506	155	5k
JMLR	3 846	75 646	40 879	580	30k
Moby Dick	10 277	105 719	22 559	231	10k

Results of SQs

JMLR

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

(top-5 from 563)

Pres. Addresses

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

(selection from top-25)

That was back in 2012

now back to 2015

SQUEEZE

Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

SQUEEZE

Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

SQUEEZE

Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

serial episodes

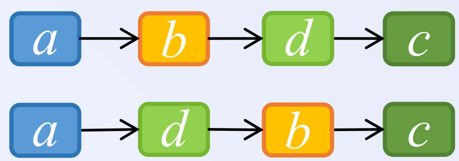
SQUEEZE

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

1) richer pattern language

serial episodes

parallel episodes



SQUEEZE

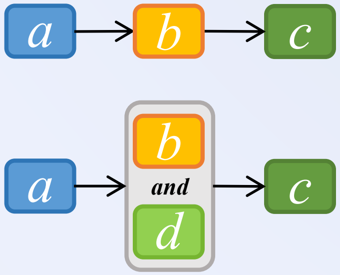
Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

serial episodes

parallel episodes



SQUEEZE

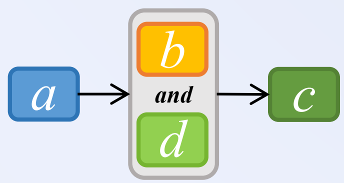
Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

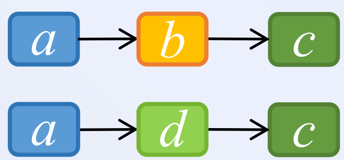
1) richer pattern language

serial episodes

parallel episodes



'choice' episodes



SQUEEZE

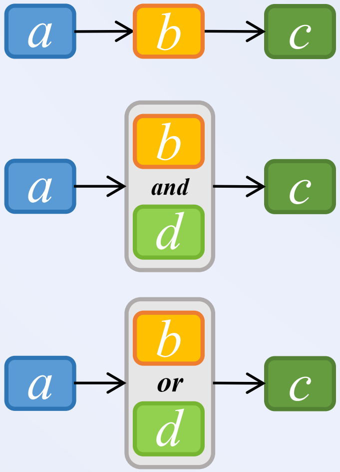
Though nice, SQS is limited
With SQUEEZE we aim to push the envelope.

1) richer pattern language

serial episodes

parallel episodes

'choice' episodes



SQUEEZE

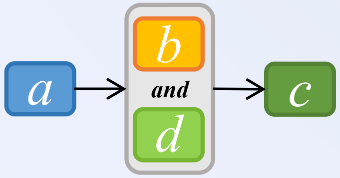
Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

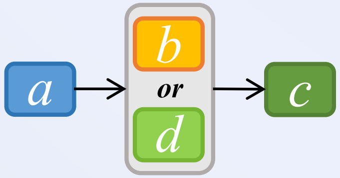
1) richer pattern language

serial episodes

parallel episodes



'choice' episodes



'stopisodes'

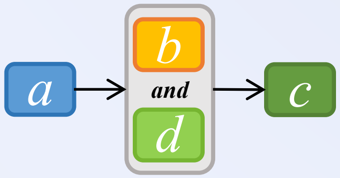
SQUEEZE

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

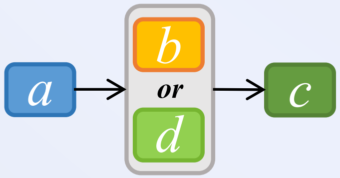
1) richer pattern language

serial episodes

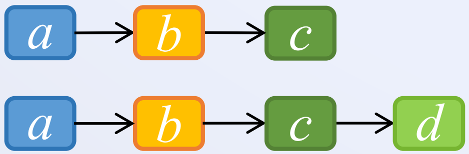
parallel episodes



'choice' episodes



'stopisodes'



SQUEEZE

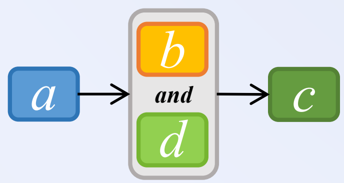
Though nice, SQS is limited

With SQUEEZE we aim to push the envelope.

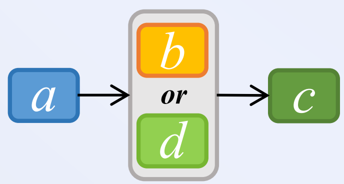
1) richer pattern language

serial episodes

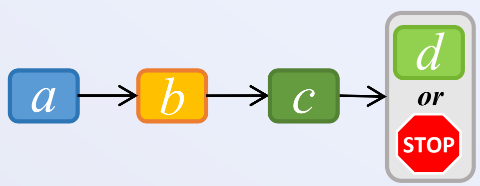
parallel episodes



'choice' episodes



'stopisodes'



SQUEEZE

Though nice, SQS is quite limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

2) better covers

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

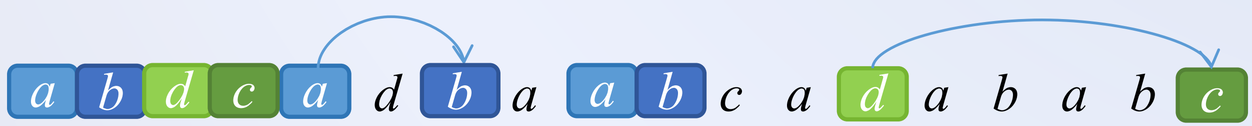
SQUEEZE

Though nice, SQS is quite limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

2) better covers

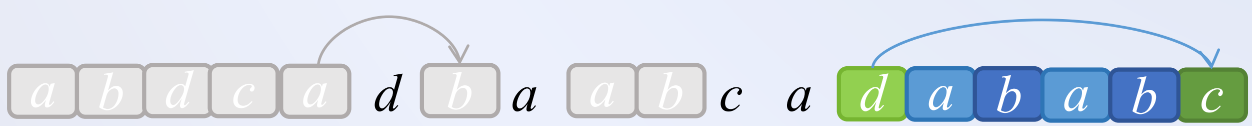


SQS: non-overlapping, non-nested, non-interleaving

SQUEEZE

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

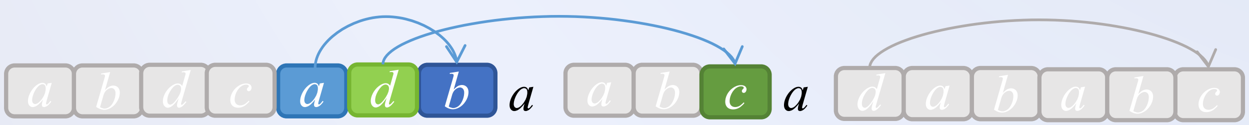
SQUEEZE

Though nice, SQS is quite limited

With SQUEEZE we aim to push the envelope.

1) richer pattern language

2) better covers



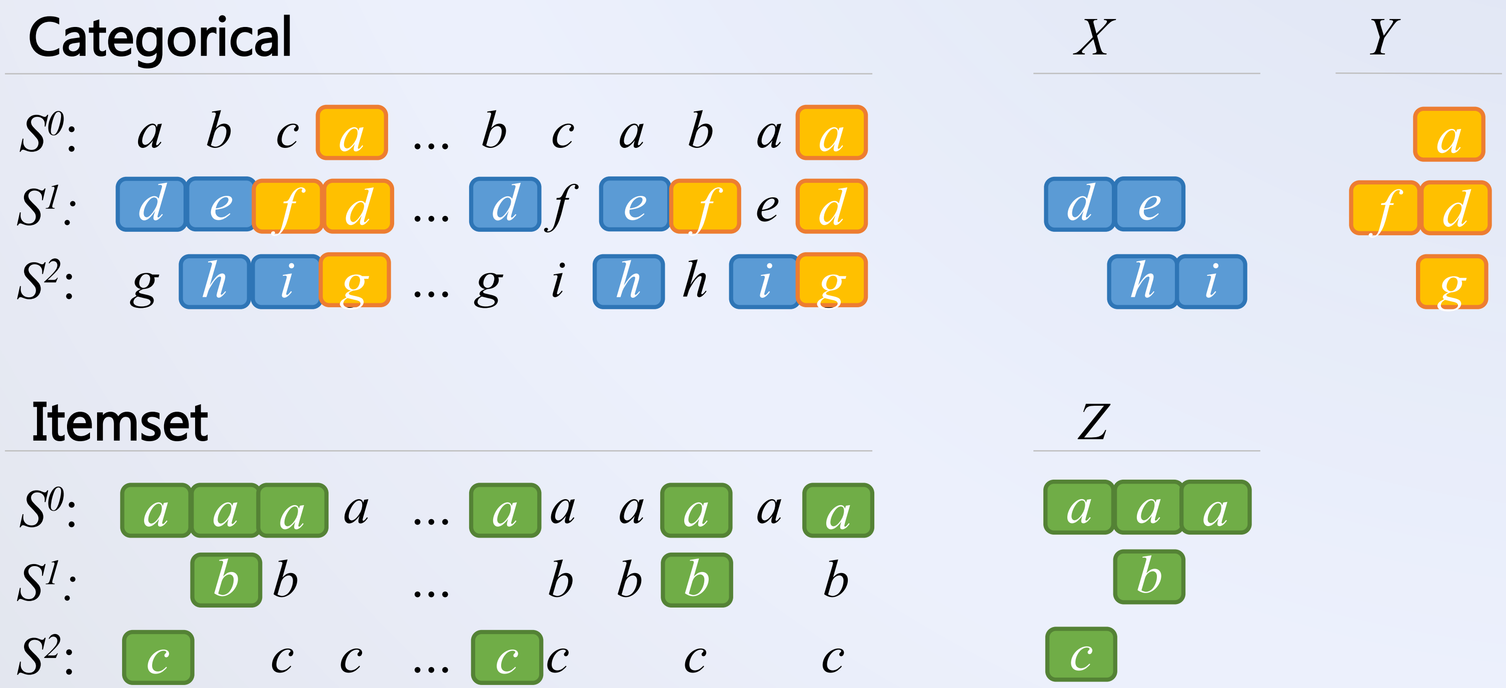
SQUEEZE: non-overlapping, non-**nesting**, non-**interleaving**

(work in progress, with Bhattacharyya)

DITTO

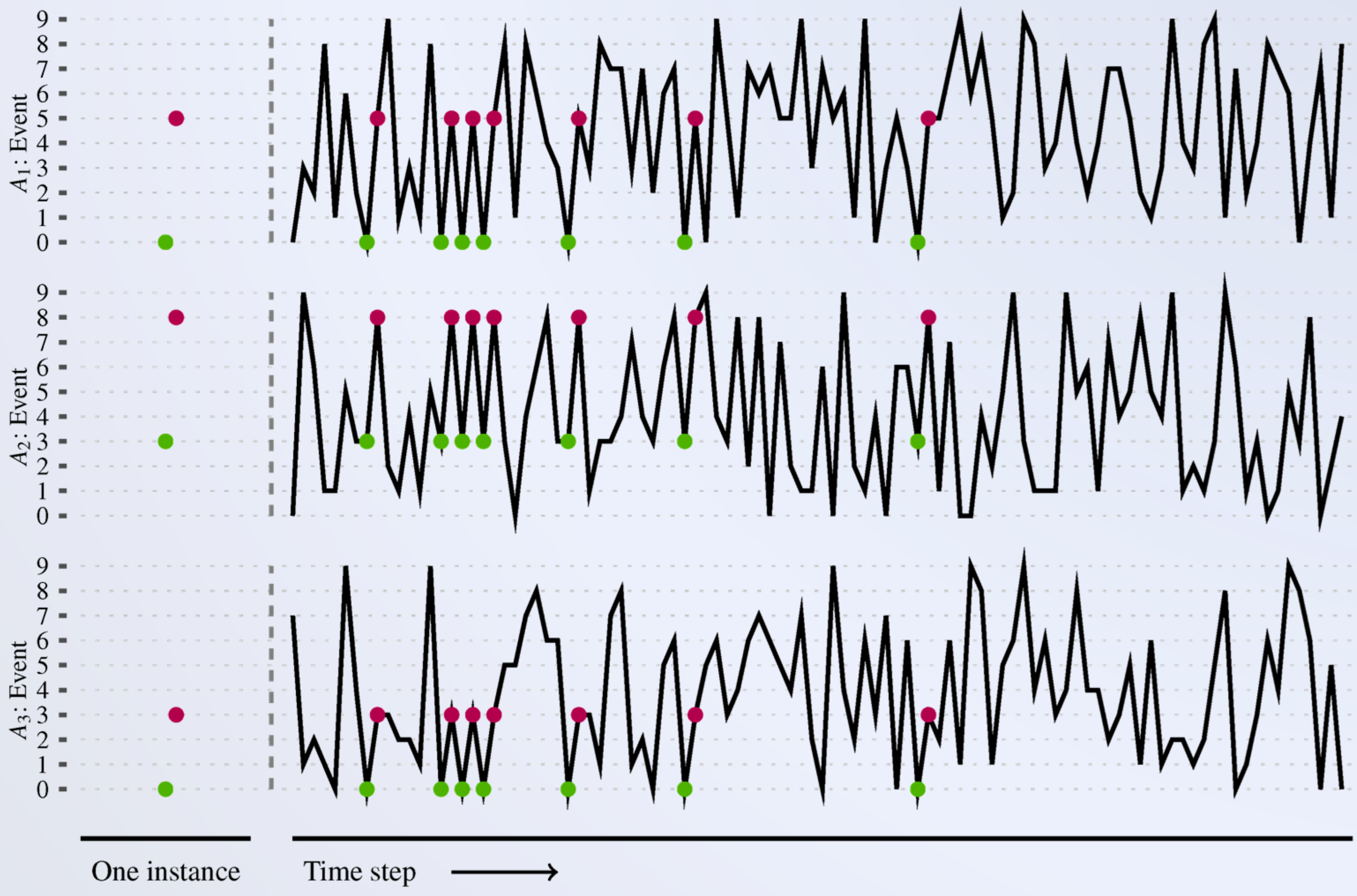
Though nice, SQS is quite limited

With DITTO we push the envelope to **multivariate** data & patterns



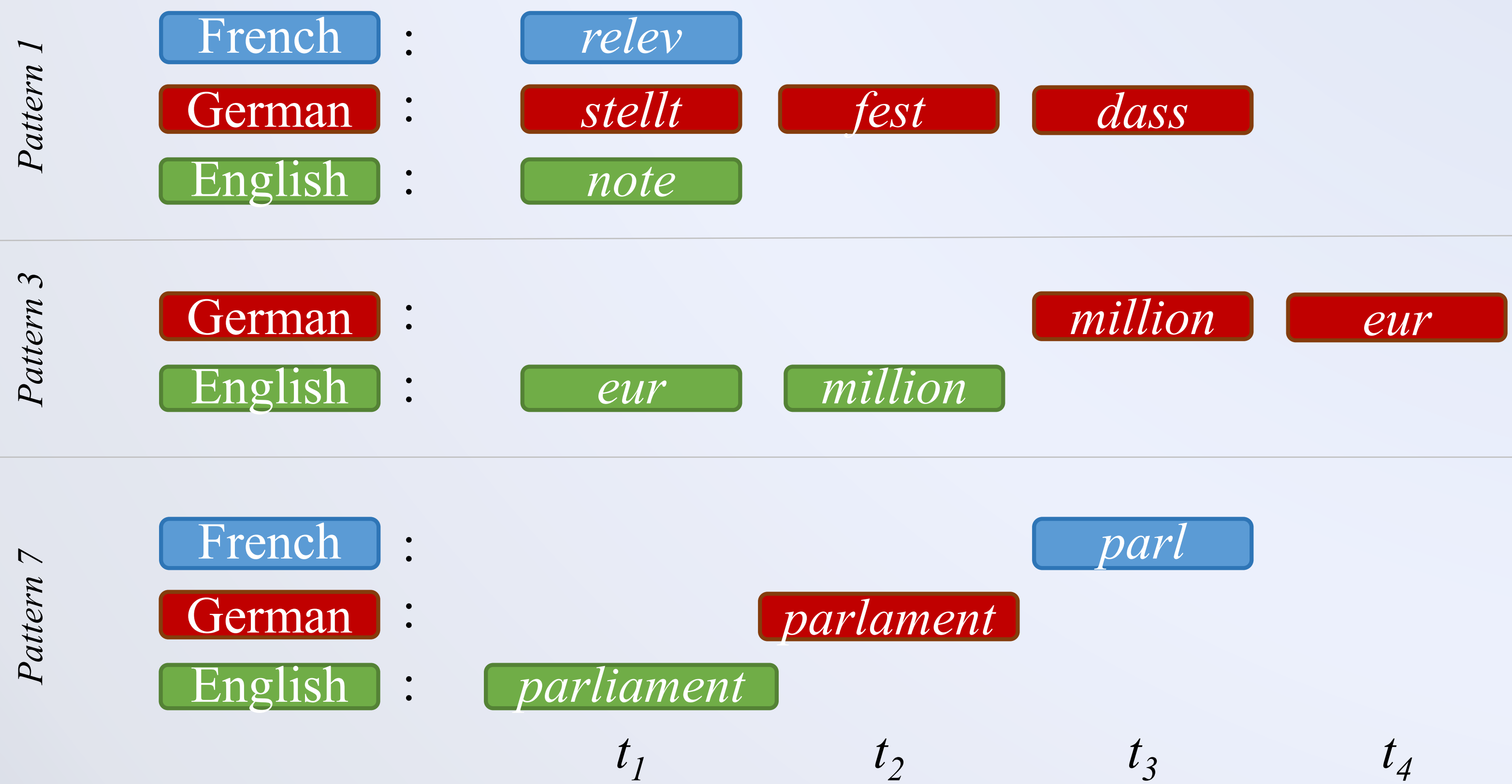
(Bertens, Vreeken & Siebes, under submission)

Ditto in Action



DITTO in Action

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,



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)