Advanced Human Machine Interaction

(Interaction) Data Analysis

Alexandre Pauchet

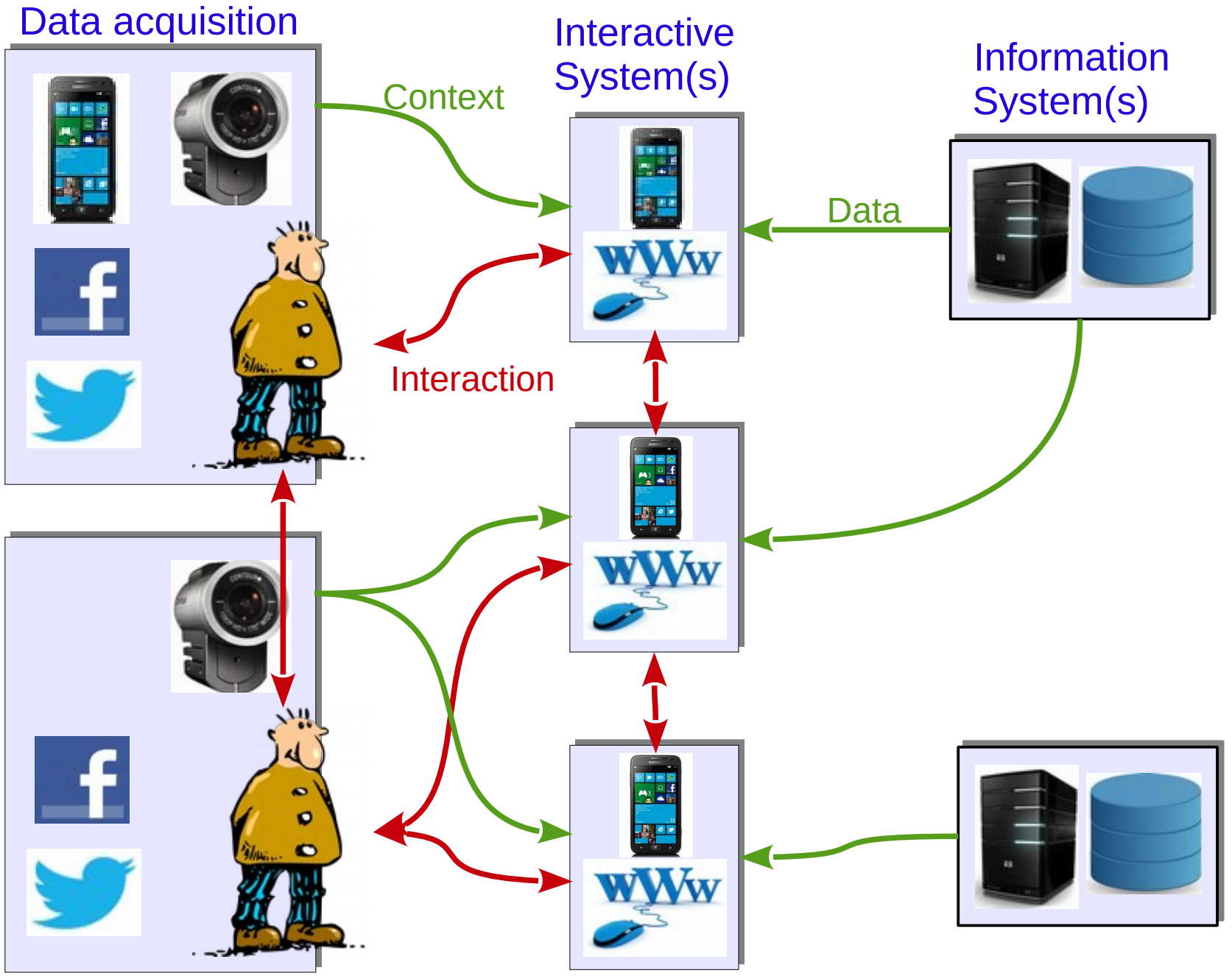
alexandre.pauchet@insa-rouen.fr - BO.B.RC18

Normandie Université

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

litis

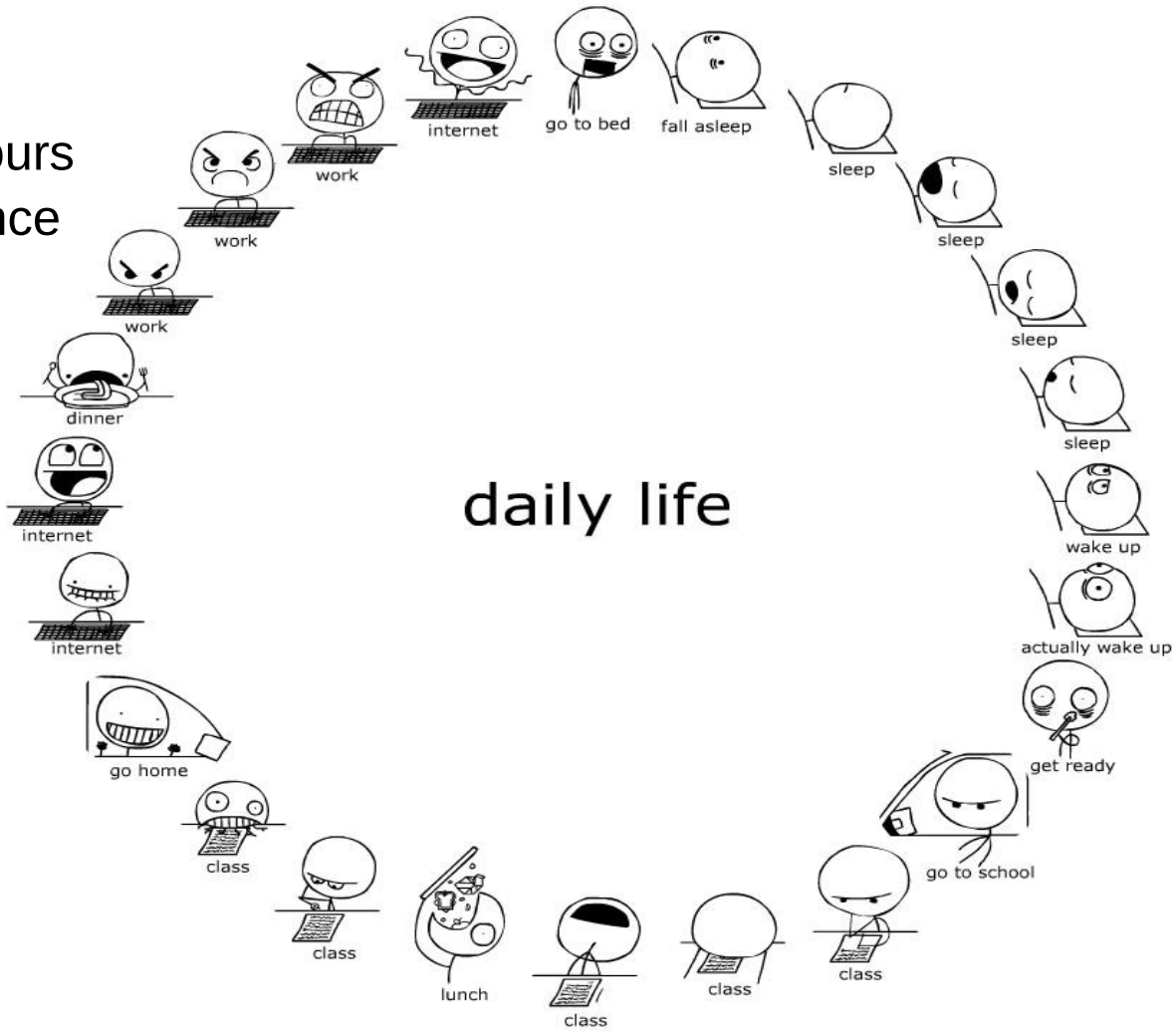
IHME/IDA: objectives



Why analyse interaction traces?

Regularities

- Users repeat actions and strategies
- Similar situations => similar behaviours
- Behaviour may evolve with experience and context



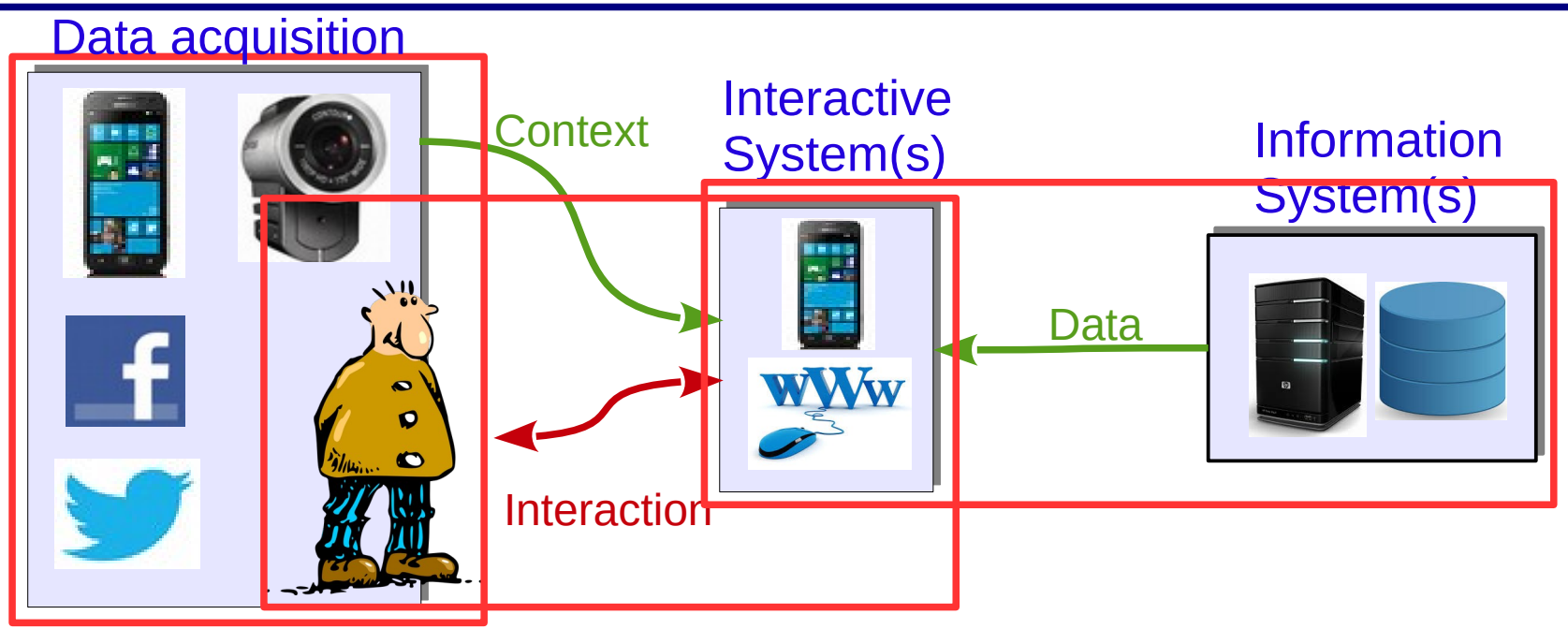
http://ennokni.deviantart.com/art/Daily-Life-117834930

Exploiting interaction traces

- Describe and compare behaviours
- Predict future actions
- Adapt an interactive system

A dataset provides a partial, instrumented trace of an interaction

Analysis of interaction data



The data to analyze can be collected from:

- External/internal sensors (context of use)
- Interaction data (log of user actions and/or activity)
- External data collected by the interactive system

Specific properties of interaction data

Structure

- Events are ordered, temporal and often dependent
- Data is nested into sessions, users and groups

Observation

- Traces may be incomplete, noisy or irregularly sampled
- Several devices may observe the same activity differently

Meaning and evolution

- An event depends on its history and context
- Users and interactive systems may change over time

Responsibility

- Interaction traces may reveal identity, habits and intentions
- Collection and reuse require explicit choices

Problem modelling

From an interaction question to an evaluation protocol

Modelling workflow

1. Interaction phenomenon
Activity or behaviour to study

2. Observable traces
What the system can record

3. Data representation
Structure given to observations

4. Target output
Result expected from the system

5. Problem class
Prediction, clustering, pattern mining, ...

6. Suitable method
Methods compatible with the representation

7. Evaluation
Evidence that the solution works

Each modelling decision constrains the following ones.

Observation unit and context (1&2)

Possible units of observation

- event or time window
- interaction episode or session
- user or group

Possible contextual information

- user and task
- state of the interactive system
- physical or social environment
- time and previous events

Example event: 12.4 s user_7 open_map puzzle_3 alone

Is this event one data instance, or one element of a larger instance?

Discrete interaction traces (3)

Event sequence

- ex: ⟨move, move, collect, open_map⟩
- Order is represented, but time may be omitted
- Representation: ordered set $\{a_1, a_2, \ldots, a_n\}$

Timestamped sequence

- ex: ⟨(move, 1.2), (collect, 3.8), (open_map, 8.1)⟩
- Order and delays are represented

Sequence of itemsets

- ex: ⟨{left_hand, right_hand}, {left_hand}, {right_hand}⟩
- Events inside an itemset share a time step or window
- Representations:
  - ordered set of item-sets

$$\{\{a_{1,1}, \ldots, a_{m,1}\}, \{a_{1,2}, \ldots, a_{m,2}\}, \ldots, \{a_{1,n}, \ldots, a_{m,n}\}\}$$

  - matrix

$$\{a_{i,j}\}_{i=1..m;\ j=1..n}$$

Multiple sequences

- 2 users/devices may produce independent traces
- Their observations may be asynchronous
- Representation: set of ordered sets

$$\{\{a_{1,1}, \ldots, a_{m1,1}\}, \{a_{1,2}, \ldots, a_{m2,2}\}, \ldots, \{a_{1,n}, \ldots, a_{mn,n}\}\}$$

Continuous & mixed interaction traces (3)

Sampled signal

- $x(t_1), x(t_2), ..., x(t_n)$
- Example: voice intensity or pointer speed

Multivariate signal

- $x(t_k) = [x_1(t_k), ..., x_m(t_k)]$
- Example: position, gaze direction or body motion

Mixed trace

- Events and sampled signals may describe the same interaction

Modelling choices

- sampling and synchronisation
- normalisation and segmentation
- discretisation

Data contains sampled observations, not continuous functions!

Target outputs and problem classes (4&5)

Next action Next-event prediction

User or activity label Classification

Similar interaction segments Similarity search or alignment

Repeated subsequences Pattern mining

Groups of users or sessions Clustering

Future numerical value Regression or forecasting

New behaviour sequence Generation

The expected output determines the problem class.

Evaluation and generalisation (7)

What should generalise?

- new sessions from known users
- new users
- future interactions

Matching evaluation split

- split by session, user or time

Parameter selection

- Choose models and parameters on development data
- Report final results on separate test data

Evaluation measure

- The metric must match the expected output

Evaluation choices belong to the problem definition.

Activity: next-action prediction

In a video game, players can perform actions from the set

A = {move, jump, collect, attack, open_map}

The game records timestamped interaction traces for several sessions and players

Formalise a system that predicts the player's next action.

Identify:
1. the interaction phenomenon
2. the observable traces and analysis unit
3. the input and expected output
4. the problem class
5. an appropriate evaluation split

Next-action prediction: correction

Phenomenon A player chooses an action according to the current situation and previous actions

Trace $\tau_{u,s} = \langle (a_1,t_1,c_1), \ldots, (a_n,t_n,c_n) \rangle$
Input $x_t = \langle (a_1,t_1,c_1), \ldots, (a_t,t_t,c_t) \rangle$
Output $y_t = a_{t+1} \in A$

Problem class Supervised next-event prediction
Multiclass classification with a sequential input

Evaluation split
- by user for unseen users
- by session for new sessions
- by time for future behaviour

Possible metrics
Accuracy, top-k accuracy or macro F1.

Discrete sequences

Similarity-based pattern extraction

Sequence alignment finds similar fragments
despite substitutions and missing or extra events

Same behaviour, different traces

Two players may follow the same strategy while producing different recorded traces.

Trace A MENU MOVE AIM SHOOT RELOAD MOVE
Trace B MOVE MENU AIM WAIT SHOOT MOVE

Question: **Do these traces contain a common interaction pattern?**

Exact matching misses patterns altered by noise or variability.

Sequence alignment

An alignment inserts gaps so that comparable events appear in the same columns.

Trace A MOVE AIM — SHOOT RELOAD
Trace B MOVE AIM WAIT SHOOT —

Match the events are identical
Substitution one event replaces another
Gap one trace contains an insertion or a deletion
The direction of insertion or deletion depends on the trace chosen as reference

Global and local alignment

Global alignment

Compares the two complete traces

Useful when both traces describe the same episode from beginning to end

LOGIN SEARCH OPEN SAVE LOGOUT
IDLE SEARCH OPEN SAVE SHARE

Local alignment

Finds the pair of fragments with the highest score

Useful when a routine appears inside longer, different episodes

SEARCH OPEN SAVE
SEARCH OPEN SAVE

Smith–Waterman computes local alignments

Scoring an alignment

Example scoring model
Match +2
Substitution −1
Gap −2

MOVE AIM — SHOOT RELOAD
MOVE AIM WAIT SHOOT —
+2 +2 −2 +2 −2
Total score: +2

- The scoring model states which variations the method should tolerate
- A strong gap penalty favours contiguous patterns
- Scores must reflect expected variability and observation noise

Activity: choosing an alignment

Use match = +2, substitution = −1 and gap = −2. Compute both scores and choose the best alignment.

Alignment 1

A B — C D

A B X C D

Alignment 2

A B C — D

A B X C D

What interpretation does each alignment propose?
Which is the best alignment?

Choosing an alignment: correction

Alignment 1

A B — C D

A B X C D

+2 +2 −2 +2 +2 = 6

Interpretation: X is an extra event.

Alignment 2

A B C — D

A B X C D

+2 +2 −1 −2 +2 = 3

Interpretation: C becomes X, then C is inserted.

Alignment 1 has the highest score.

The scoring model selects the preferred explanation.

Smith–Waterman local alignment

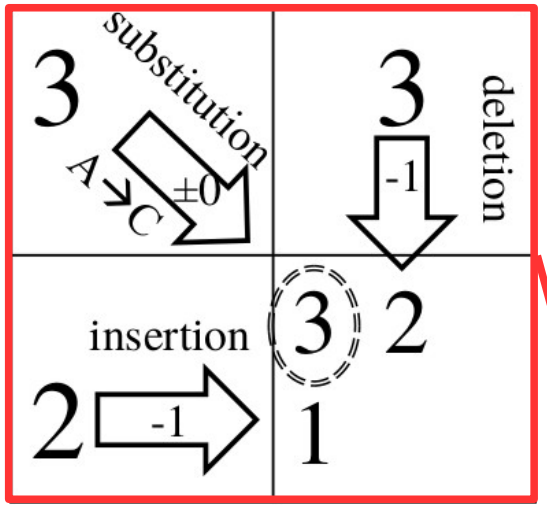
- A matrix stores the best local score ending at each pair of positions.
- Previously computed prefixes provide the next value.
- A zero starts a new local alignment.

$$\mathbf{H[i,j] = \max}\begin{cases} 0 \\ H[i-1,j-1] + \mathrm{sub}(x_i,y_j) \\ H[i-1,j] + \mathrm{gap} \\ H[i,j-1] + \mathrm{gap} \end{cases}$$

Time and memory: O(mn)

Dynamic programming calculation

Smith & Waterman



$$
\begin{aligned}
t[-1,-1] &= 0, \\
t[i,-1] &= 0, \\
t[-1,j] &= 0, \\
t[i,j] &= max\begin{cases} t[i-1,j-1]+Subs(x[i],y[j]), \\ t[i-1,j]+Del(x[i]), \\ t[i,j-1]+Ins(y[j]), \\ 0 \end{cases}
\end{aligned}
$$

sub(x,x) = 1
sub(x,z) = 0
Ins = del = -1

	A	C	G	T	C	G	A	C	G
A	1	0	0	0	0	0	1	0	0
C	0	2	1	0	1	0	0	2	1
T	0	1	2	2	1	1	0	1	2
C	0	1	1	2	3	2	1	1	1
A	1	0	1	1	2	3	3	2	1
C	0	2	1	1	2	2	3	4	3
G	0	1	3	2	1	3	2	3	5

Remarks:

- Subs/Del/Ins have negative values
- The value at (i,j) position in table t only depends on the 3 adjacent positions

Computing one matrix cell

X = A B C
Y = A X C

	A	X	C
	0	0	0	0
A	0	2	0	0
B	0	0	1	0
C	**0**	**0**	**0**	**3**

For H[3,3], compare C with C

restart: 0
diagonal: 1 + 2 = 3
up: 0 − 2 = −2
left: 0 − 2 = −2

H[3,3] = 3

Each cell needs only its diagonal, upper and left neighbours.

Traceback and extracted pattern

Traceback

- Start from the highest score in the matrix
- Follow the predecessor that produced each value
- Stop when the score reaches zero

Result for the example

A B C

A X C

score = 3

The substitution B/X is tolerated because the surrounding matches compensate for its penalty

	A	C	G	T	C	G	A	C	G
A	1	0	0	0	0	0	1	0	0
C	0	2	0	0	1	0	0	2	0
T	0	0	0	1	0	0	0	0	0
C	0	1	0	0	2	0	0	1	0
A	1	0	0	0	0	0	1	0	0
C	0	2	0	0	1	0	0	2	0
G	0	0	3	1	0	2	0	0	3

	A	C	G	T	C	G	A	C	G
A	1	0	0	0	0	0	1	0	0
C	0	2	1	0	1	0	0	2	1
T	0	1	2	2	1	1	0	1	2
C	0	1	1	2	3	2	1	1	1
A	1	0	1	1	2	3	3	2	1
C	0	2	1	1	2	2	3	4	3
G	0	1	3	2	1	3	2	3	5

The matrix gives a score; traceback gives the aligned fragments.

Activity: formalising pattern extraction

A dataset contains discrete interaction traces. We want to extract pairs of locally similar behavioural fragments.

1 Observable data What is recorded?
2 Representation How do we represent one trace and the dataset?
3 Expected output What precisely constitutes an extracted pattern?
4 Parameters Which modelling choices influence the result?
5 Evaluation What evidence would show that the extraction is useful?

Pattern extraction: formal model

Alphabet A finite set of interaction events A

Dataset $T = \{t_1, \ldots, t_r\}$, with each $t_i \in A^*$

Parameters substitution scores, gap penalty, minimum score θ and optional minimum length ℓ

Output pairs of subsequences $x \subseteq t_i$ and $y \subseteq t_j$, their alignment and score, with score $\geq \theta$

Problem class similarity-based local pattern extraction

Method pairwise local sequence alignment, followed by selection or grouping of alignments

Possible evaluation

- precision and recall against annotated patterns
- stability when traces or scores change
- runtime and number of redundant patterns

The parameters express assumptions about behavioural variability and sensor noise

From alignments to frequent patterns

Pairwise local alignments

(t_1, t_2) SEARCH OPEN

(t_1, t_4) SEARCH OPEN SAVE

(t_2, t_5) SEARCH OPEN

The result may contain many overlapping or redundant candidates.

Frequency questions

- How many times does a pattern occur?
- In how many traces or users does it occur?
- Must occurrences match exactly?
- How should similar candidates be grouped?

Frequency requires a precise definition of an occurrence and its population

Discrete sequences

Frequency-based pattern extraction

Suffix trees index exact contiguous patterns and their occurrences.

What counts as the same pattern?

Substring events are consecutive: A B

Example occurrence: X A B Y

Subsequence events stay ordered but gaps are allowed: A … B

Example occurrence: X A C D B Y

Approximate pattern substitutions or gaps are accepted according to a score

Sequence alignment addresses this third case

Suffix trees index exact substrings.

Support may count occurrences, traces or users.

Suffix trie for one trace

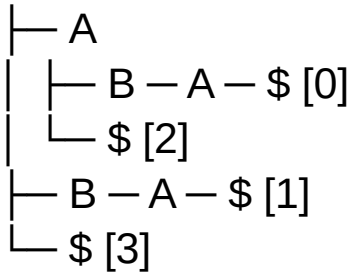
Trace: A B A $

Suffixes and starting positions

0 A B A $
1 B A $
2 A $
3 $

The terminator $ makes every suffix end at a distinct leaf

root



A root-to-leaf path spells one suffix.
The leaf label stores its starting position

Activity: constructing a suffix trie

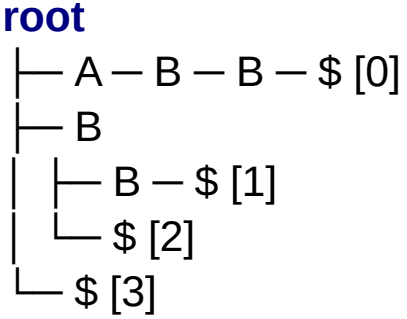
Trace: A B B $

1. List every suffix and its starting position.
2. Insert the suffixes into a trie.
3. Which substring occurs more than once?
4. How can you identify that repetition in the trie?

Three minutes in pairs

Suffix trie: correction

0 A B B $
1 B B $
2 B $
3 $



B occurs twice.

Its node has two descendant leaves: [1] and [2].

From suffix trie to suffix tree

Compaction

A non-branching path becomes a single labelled edge

The compact tree keeps the same suffixes and branching points

For A B A $

root

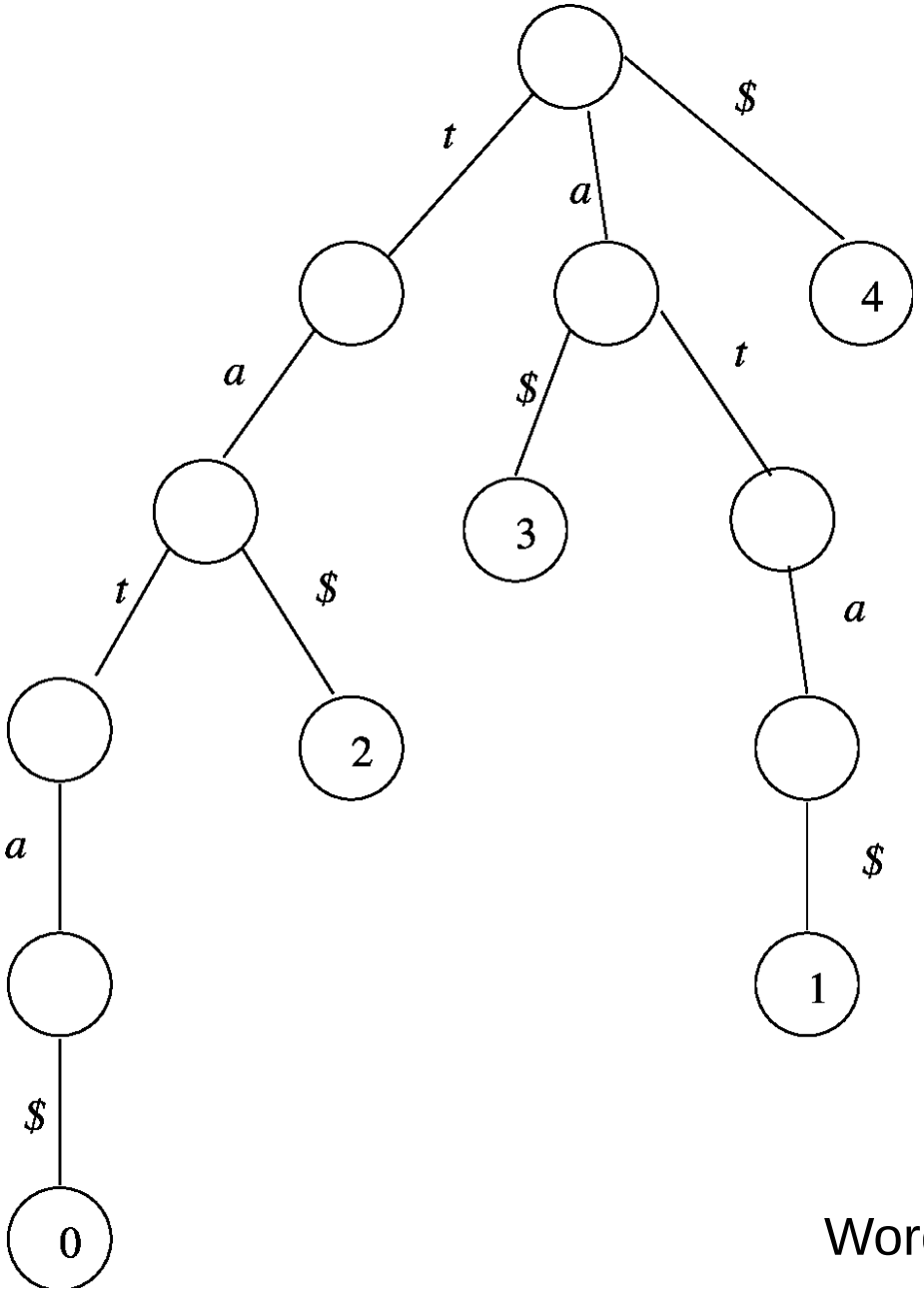
```
├── A
│   ├── B A $ [0]
│   └── $ [2]
├── B A $ [1]
└── $ [3]
```

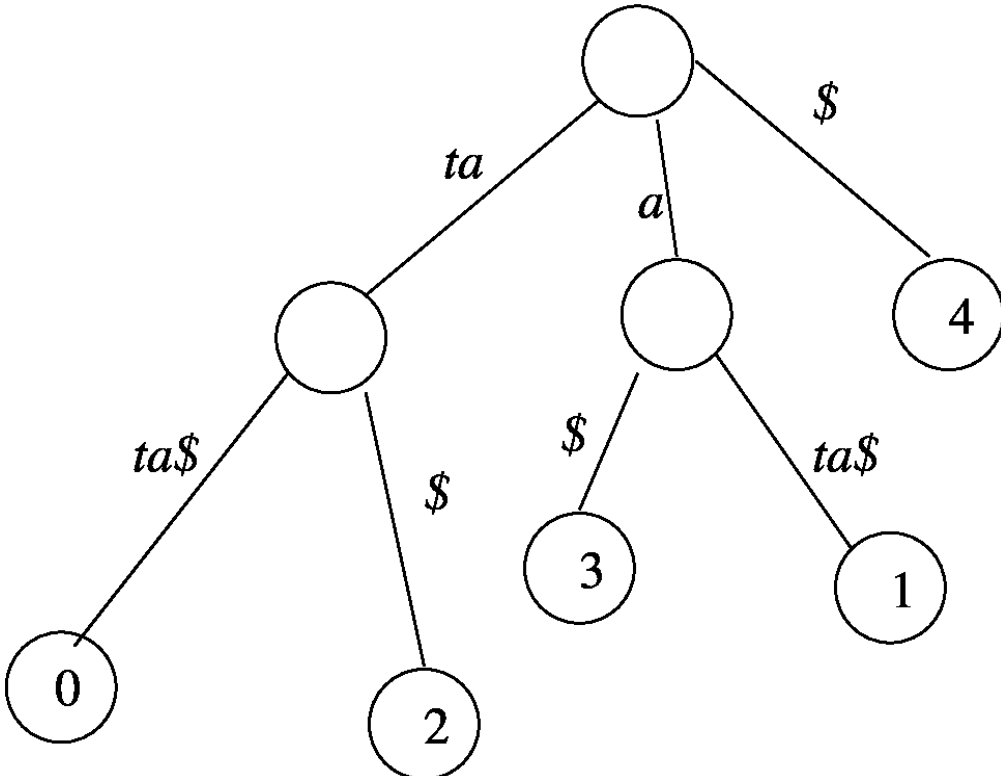
Leaf labels

[p] the suffix starts at position p

Positions use zero-based indexing in these slides

Example of suffix trie





Word: “*tata*”

Generalised suffix tree

T_1 = A B A B $1
T_2 = X A B C $2

Unique terminators prevent suffixes from different traces from merging at their ends.

Path A B

descendant leaves: (1,0), (1,2), (2,1)

occurrence count = 3

trace support = 2

Direct construction

1 Append a unique terminator to each trace.

2 For each position p, insert the suffix starting at p.

3 Label its leaf with (trace, position).

4 Compress maximal non-branching paths.

For total length N, this direct construction may take $O(N^2)$ time. The compact tree occupies O(N) space.

Linear-time construction algorithms exist, but their implementation is outside this course.

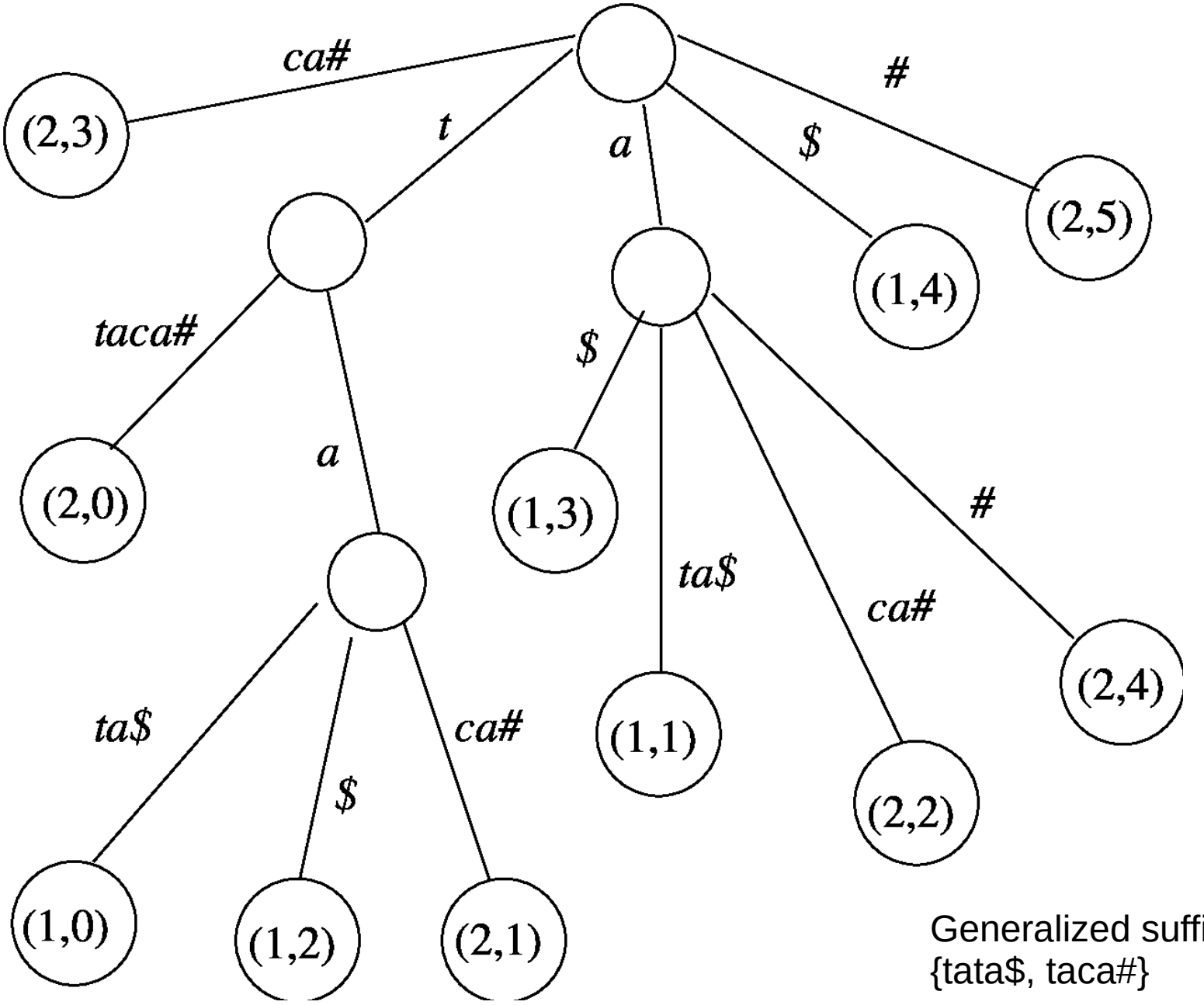
Leaf labels

(k,p) the suffix starts at position p in trace k

Example

(2,0) denotes the complete suffix beginning with the first event of trace 2.

Generalized suffix tree (multiple words)



Generalized suffix tree for {tata$, taca#}

Activity: querying repeated patterns

T_1 = A B C A B D

T_2 = X A B C Y

1 What is the longest substring present in both traces?

2 Where does it start? Use zero-based positions.

3 Give its occurrence count and trace support.

4 Does the shorter pattern A B occur more often?

Repeated patterns: correction

Longest shared substring

A B C

positions: (1,0), (2,1)
occurrence count = 2
trace support = 2

More frequent shorter substring

A B

positions: (1,0), (1,3), (2,1)
occurrence count = 3
trace support = 2

Longest and most frequent describe different target outputs.

Case study: recurrent navigation paths

Interaction phenomenon

Users repeatedly follow similar paths through an online health catalogue.

Target output

Repeated navigation substrings with their positions and supports.

Observable traces

Page requests, timestamps and session information.

Suitable method

Generalised suffix tree and longest repeated substrings.

Representation

One resource identifier per page and one terminated sequence per episode.

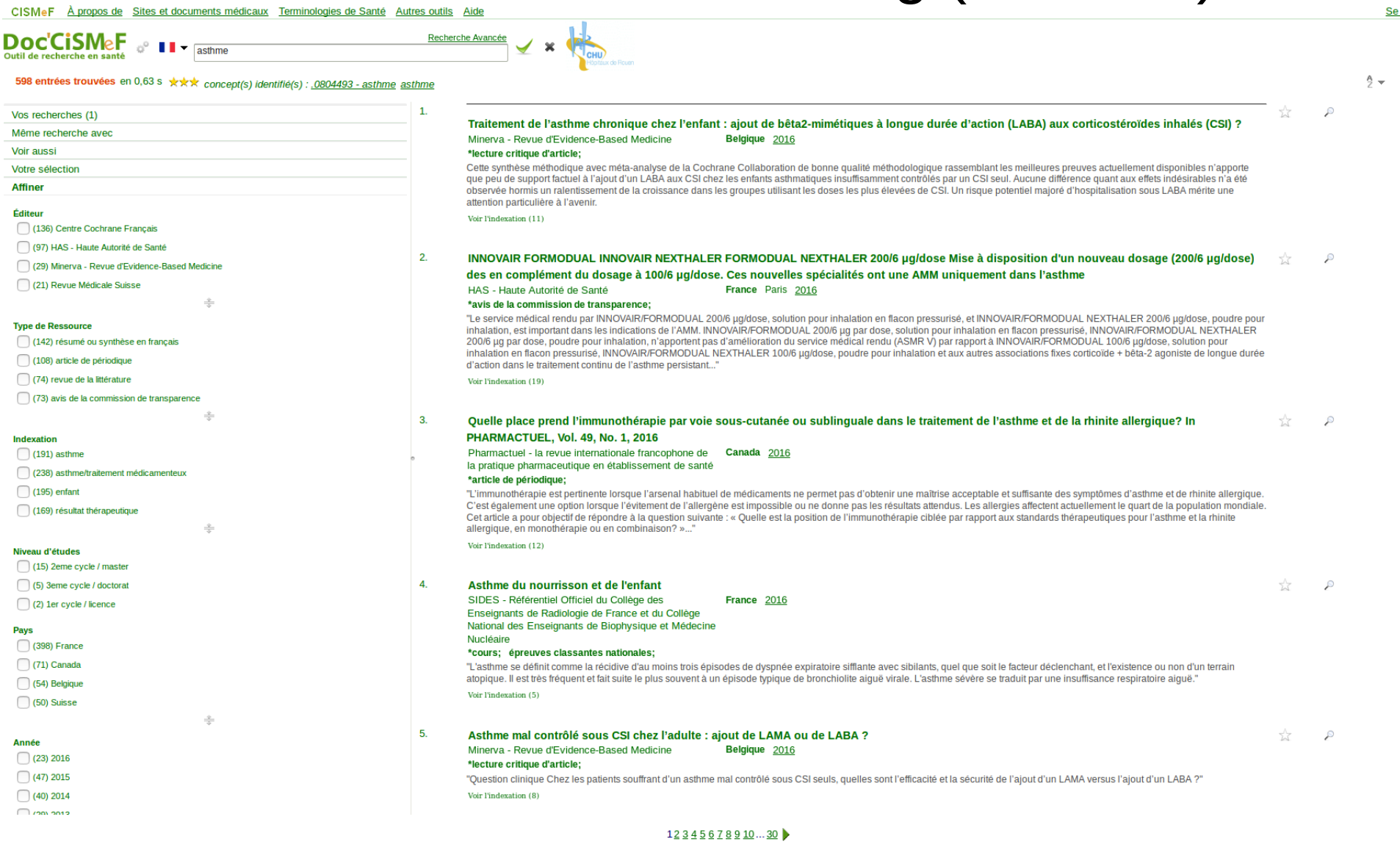
Interpretation limits

The result depends on episode boundaries and resource encoding. Exact matching does not tolerate noisy or alternative paths.

CISMeF example

Application: CISMeF

- Extraction of recurrent behaviors in the navigations within an online health catalog (CISMeF)



Application: CISMeF

- **Data preparation**
  - Episode extraction: IP + semantic distance between documents + time between requests
  - Resource identification: unique ID + delimiter

    example of session: /59451/ /303901/ /170702/

- **Recurrent pattern extraction**
  - Generalized suffix tree
  - Longest repeated substrings

CM1 synthesis

Similarity-based extraction

- Sequence alignment tolerates substitutions and gaps.
- The scoring model defines acceptable variations.
- Frequency requires an additional grouping step.

Frequency-based extraction

- Suffix trees index exact contiguous patterns.
- Occurrence count and trace support answer different questions.
- Segmentation and encoding determine the patterns found.

Method choice follows the formalisation of the interaction problem.

Next: richer representations, evaluation and other mining methods.

CM2: richer interaction data

How can we preserve simultaneity?

How can we discover repeated behaviour while preserving order?

How can raw signals become analysable data?

The representation determines which patterns a method can find.

The representation is part of the model

Modelling step	Guiding question
Interaction phenomenon	What behaviour or situation do we want to study?
Observable traces	What can the system actually record?
Data representation	**Which structure must the data preserve?**
Target output	What should the result describe?
Problem class	What computational problem does this define?
Suitable method	Which method solves that problem?
Evaluation	At which level should the result be validated?

Today: what structure must the data representation preserve?

order simultaneity time distance

Activity: what information must be preserved?

Trace A	Trace B	Trace C
00:08 search speech 00:08 filter touch 00:15 open touch *Multimodal events*	login search open Save *Interaction actions*	(x1, y1, t1) (x2, y2, t2) ... (xn, yn, tn) *GPS samples*

Identify the information that must not be lost when preparing each trace for analysis.

Question 1	Question 2	Question 3
Which observations belong together?	Does their order matter?	Is the distance between values meaningful?

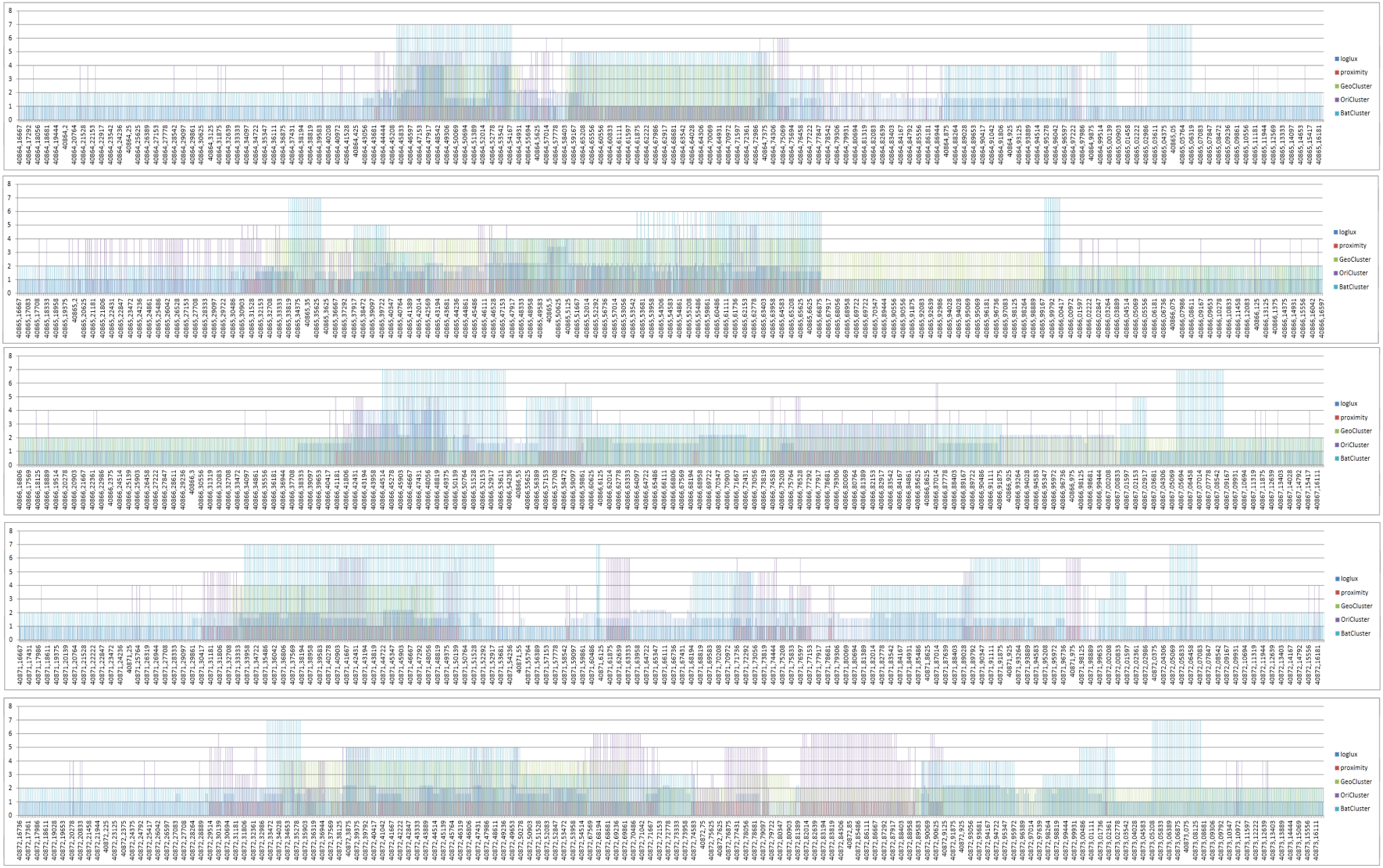
Three structures to preserve

Observable trace	Relevant structure	Possible representation
Multimodal events	Simultaneity and source	Sequence of itemsets
Interaction actions	Order, repetitions and gaps	Discrete sequence
GPS samples	Position, time and distance	2D time series or trajectory

Algorithm selection comes after the representation choice.

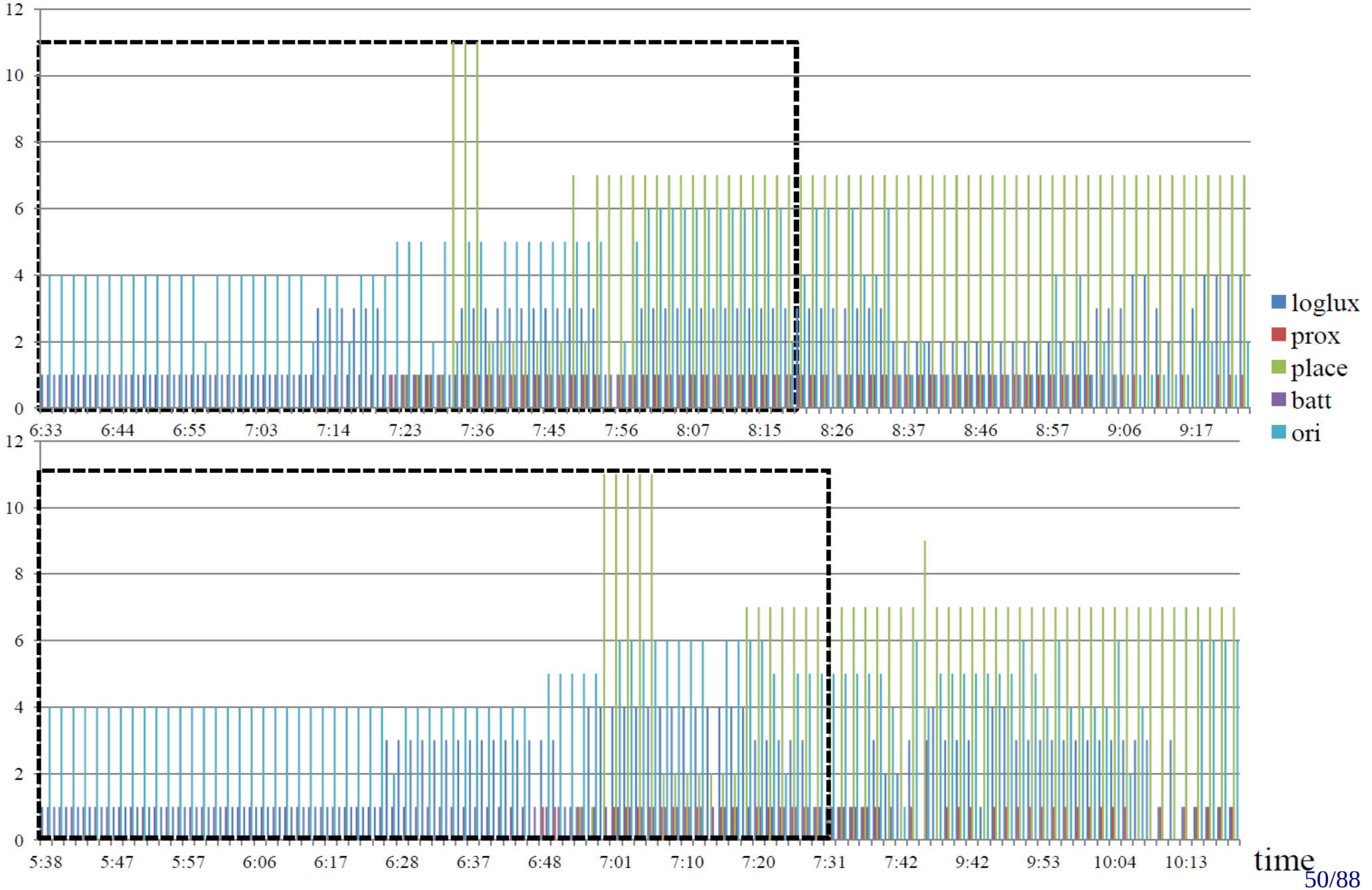
Discrete sequences of item-sets
Pattern extraction
Similarity-based approach

Routine extraction by matrix alignment

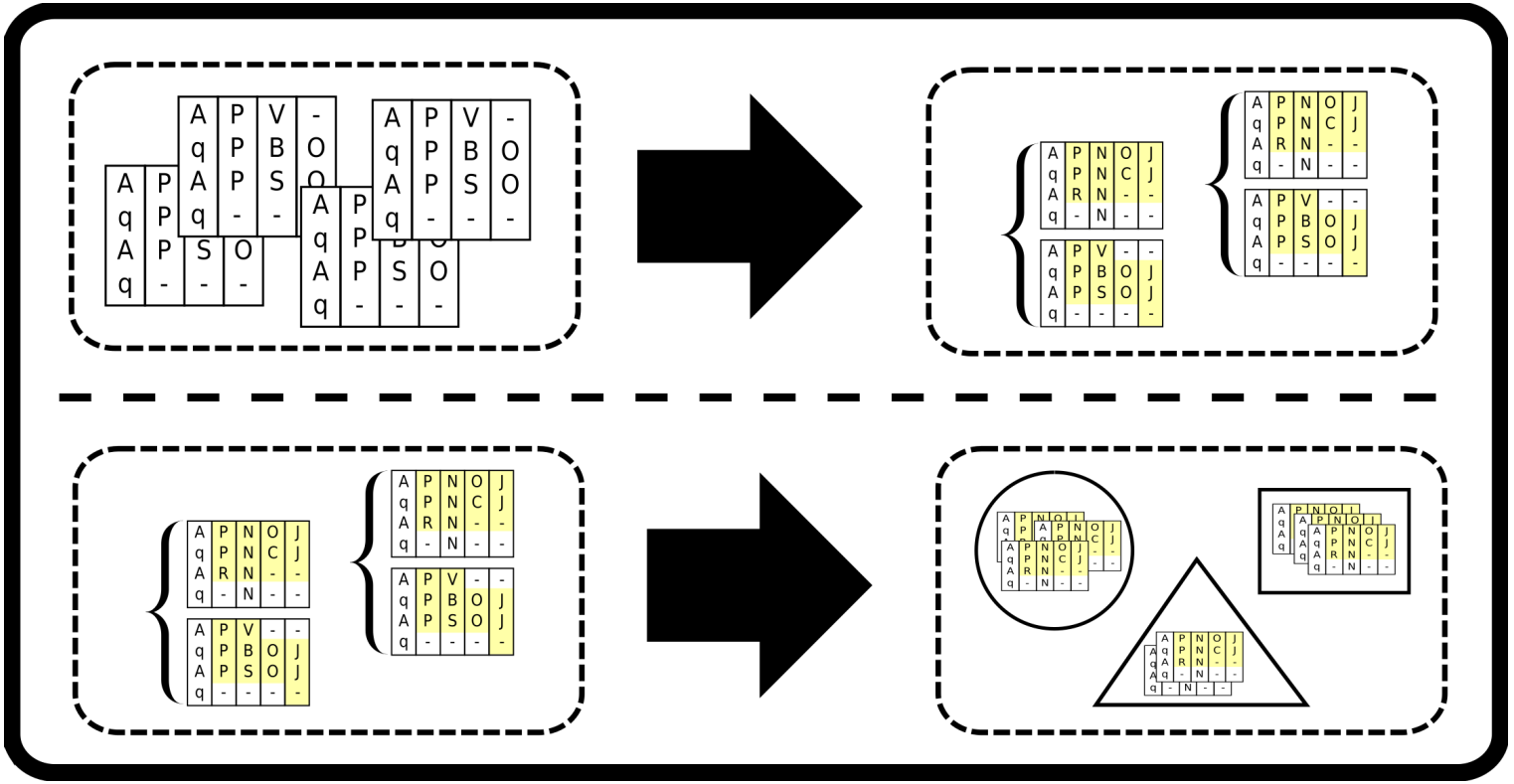


Rick Moritz (PhD thesis), Orange labs - Meylan

Routine extraction by matrix alignment



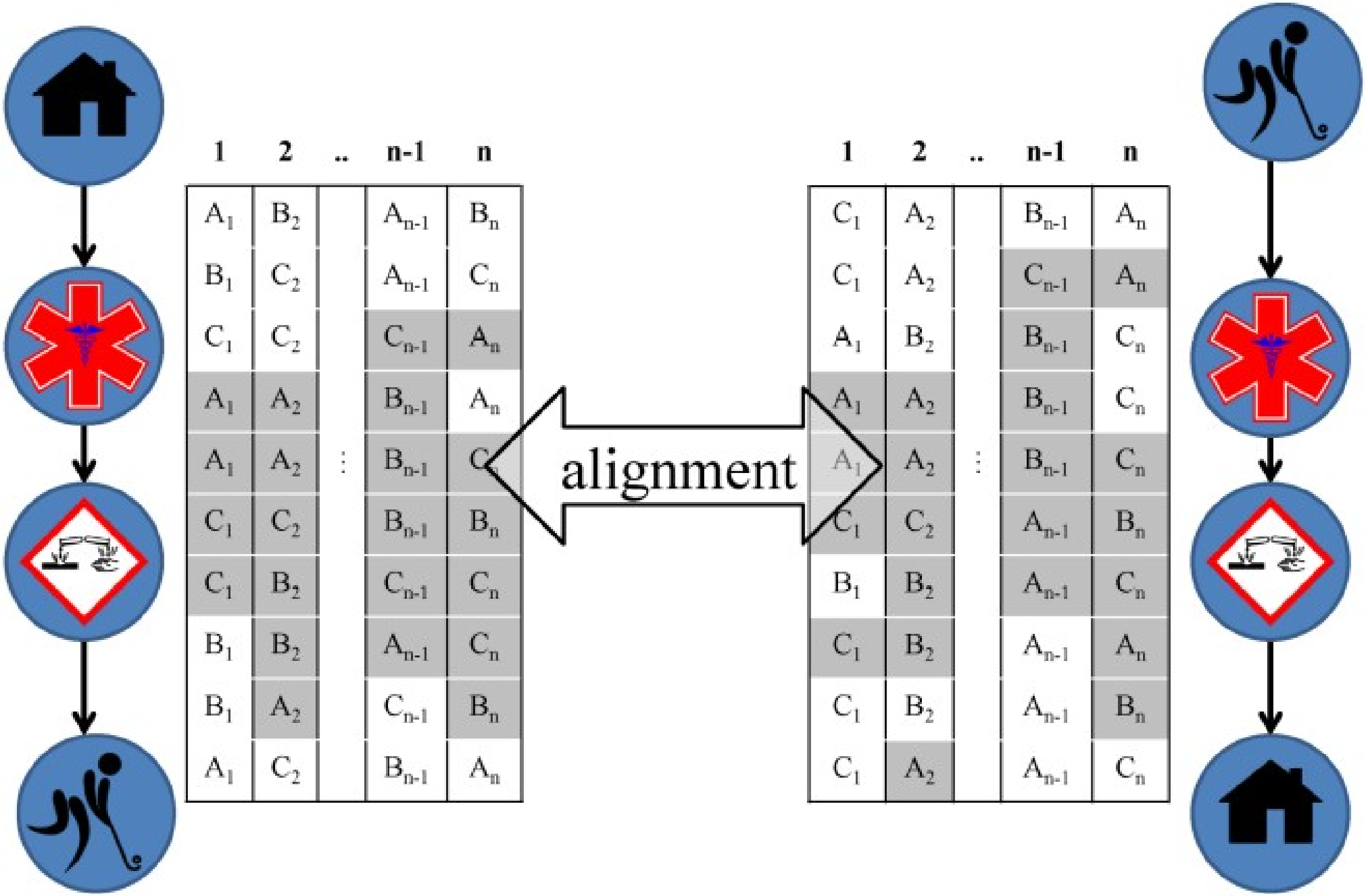
Approach (similar to sequences)



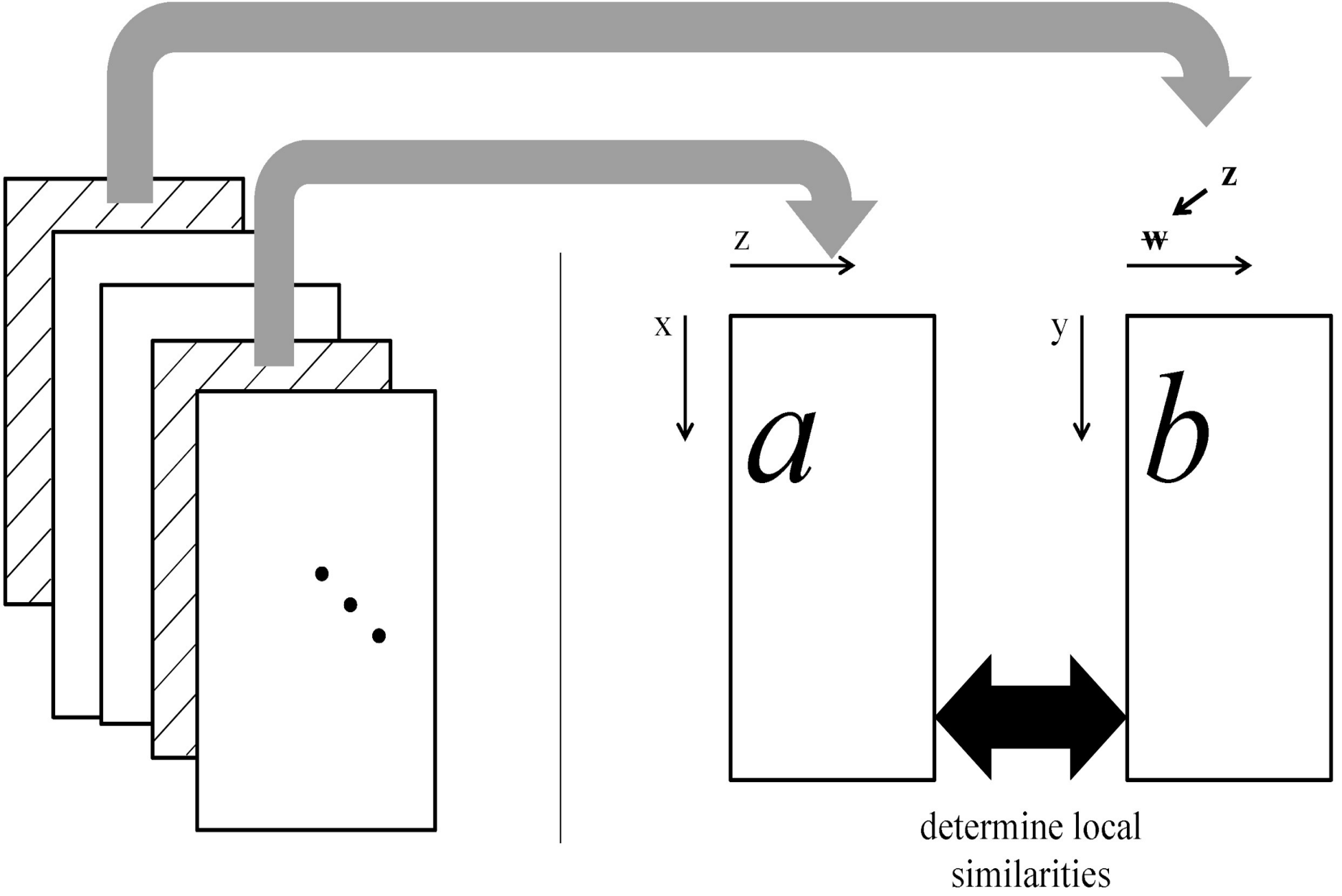
Decomposition in two separated steps

- Extraction of (pair of) interaction patterns
- Clustering of interaction patterns

Routine extraction by matrix alignment

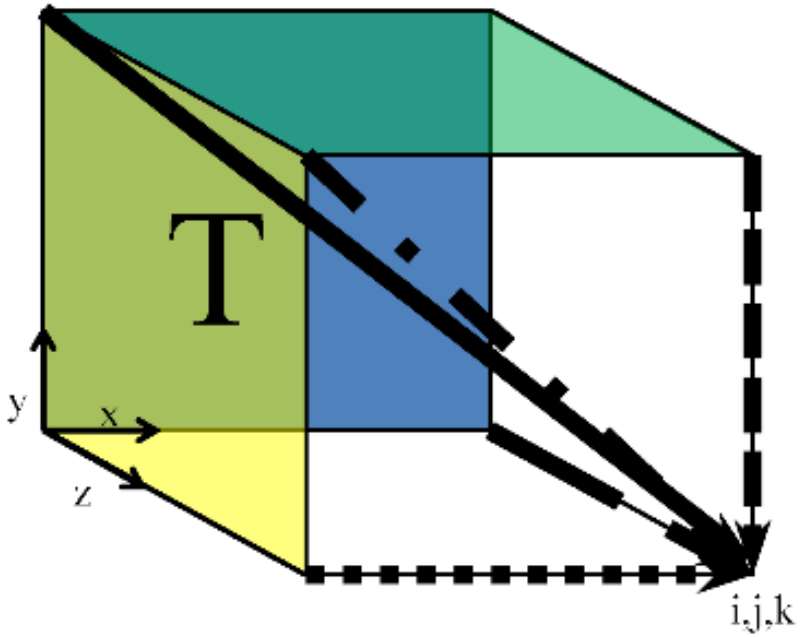


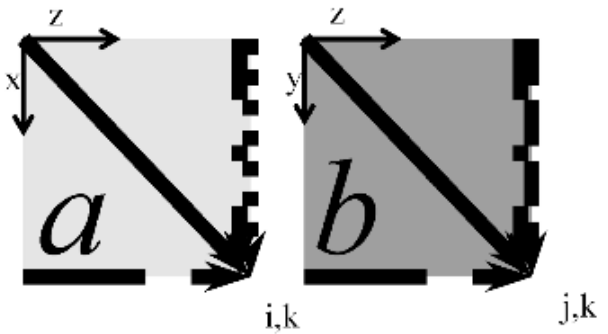
Routine extraction by matrix alignment

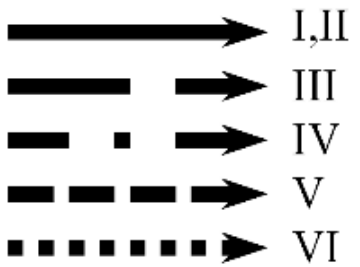


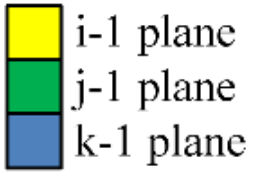
Routine extraction by matrix alignment

$$\mathrm{sim}(a,b) = \max \left(\begin{array}{cc} 0 & \\ \mathrm{sim}(a_{1,1},b_{1,1}) + \mathrm{sim}(a\,[0,0]\uparrow, b\,[0,0]\uparrow) + \mathrm{sim}(a\,[0,1]\leftarrow, b\,[0,1]\leftarrow) & \text{(I)} \\ \mathrm{sim}(a_{1,1},b_{1,1}) + \mathrm{sim}(a\,[1,0]\uparrow, b\,[1,0]\uparrow) + \mathrm{sim}(a\,[0,0]\leftarrow, b\,[0,0]\leftarrow) & \text{(II)} \\ \mathrm{sim}(a_{0,1},b_{0,1}) + \mathrm{sim}(a\,[0,0]\uparrow, b\,[0,0]\uparrow) & \text{(III)} \\ \mathrm{sim}(a_{1,0},b_{1,0}) + \mathrm{sim}(a\,[0,0]\leftarrow, b\,[0,0]\leftarrow) & \text{(IV)} \\ \mathrm{sim}(a_{0,0},b_{1,0}) + \mathrm{indel}(b[0,0]) + \mathrm{sim}(a[0,0]\leftarrow, b[1,0]\leftarrow) & \text{(V)} \\ \mathrm{sim}(a_{1,0},b_{0,0}) + \mathrm{indel}(b[0,0]) + \mathrm{sim}(a[1,0]\leftarrow, b[0,0]\leftarrow) & \text{(VI)} \end{array} \right)$$



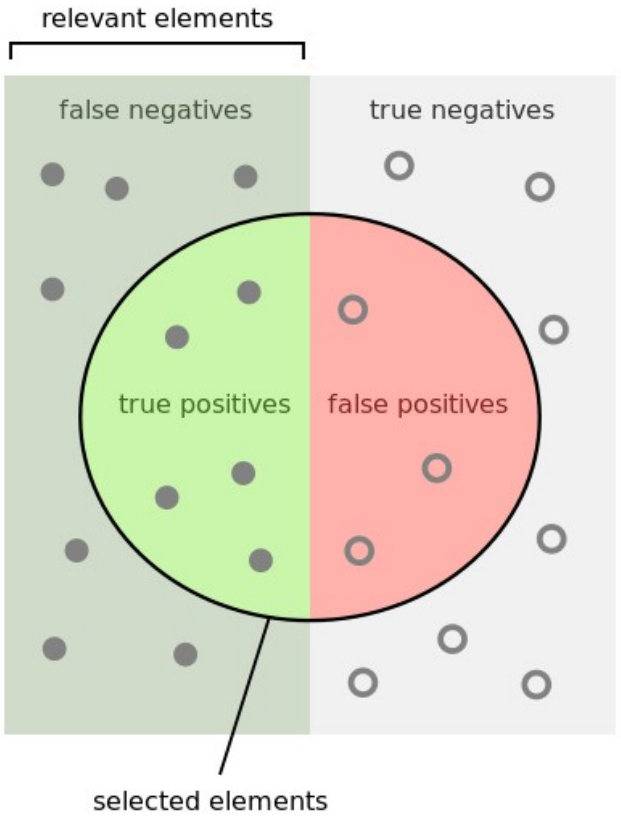


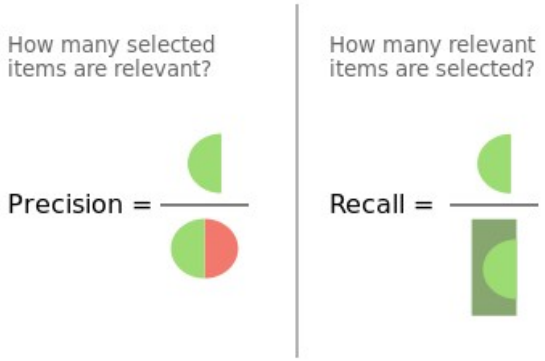




i lies on the x-axis
j lies on the y-axis
k lies on the z-axis

Evaluation: usual measures





- **Model evaluation**:
  - Precision

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

  - Recall

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

  - F-measure $F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

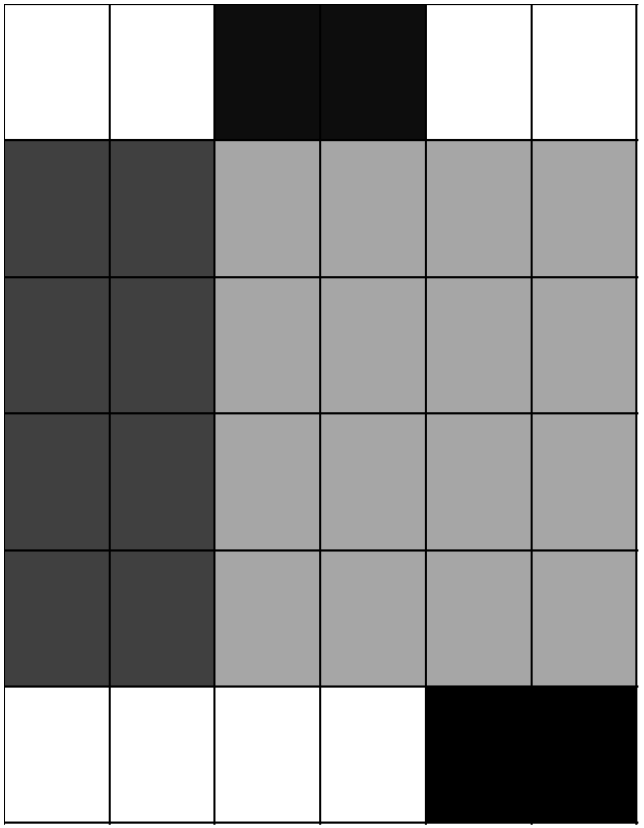
- **Remarks**:
  - Unbalanced classes!
  - Evaluation by class

https://en.wikipedia.org/wiki/Precision_and_recall

Routine extraction by matrix alignment

Evaluation

- Data and ground truth
- Computation time
- Precision and recall for each pattern alignment
- Alignment size
- Number of alignments



Total pattern cells: 24
Total aligned cells: 20
Aligned pattern cells = 16
Precision = 16/20 = 4/5
Recall = 16/24 = 2/3
Size ratio = 20/24 = 5/6

(False Positives) *(True Negatives)*

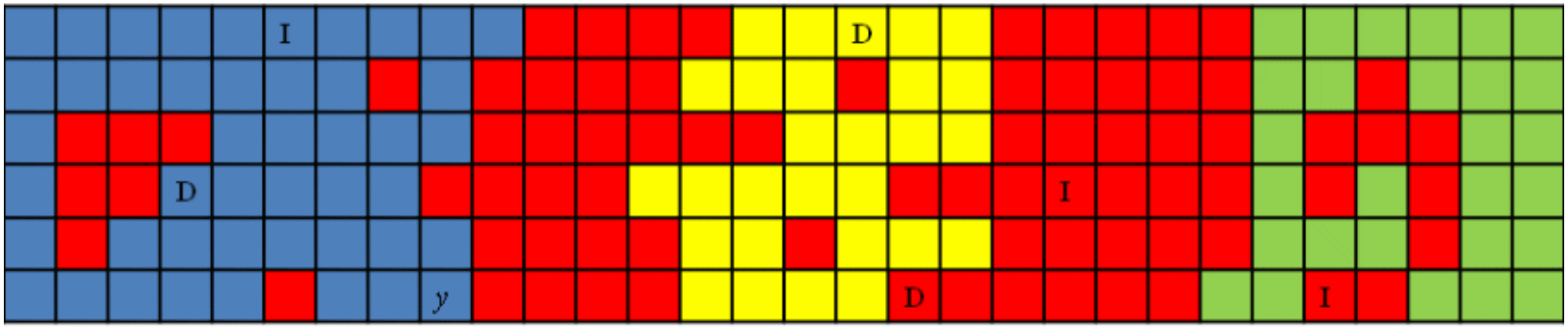
Aligned cells outside pattern | Unaligned non-pattern cells

Unaligned pattern cells | Aligned pattern cells

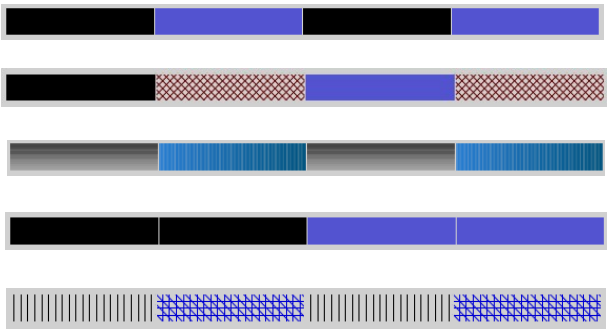
(False Negatives) *(True Positives)*

Routine extraction by matrix alignment

- Synthetic data generation



- Scenarios:
  1) Regular activity, no noise
  2) Regular activity, noise between patterns
  3) Noisy patterns
  4) Irregular pattern apparition
  5) Faulty sensors (¾ random data)



Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
$\frac{\#\text{alignments}}{\#\text{pairs of pat.}}$	0.31 ± 0.26	0.69 ± 0.29	0.041 ± 0.054	0.31 ± 0.26	0.22 ± 0.32
precision	0.54 ± 0.22	0.77 ± 0.20	0.13 ± 0.06	0.52 ± 0.24	0.54 ± 0.18
recall	1.00 ± 0.00	1.00 ± 0.00	0.56 ± 0.19	1.00 ± 0.01	0.20 ± 0.06
$\frac{\text{alignment size}}{\text{pattern size}}$	5.66 ± 4.48	2.45 ± 2.51	6.72 ± 3.96	6.65 ± 6.08	1.60 ± 0.24

Sequences of itemsets

Several observations may describe the same interaction step

From event logs to sequences of itemsets

Time	Event	Source
00:00	login	keyboard
00:08	search	speech
00:09	filter	touch
00:15	open	touch
00:42	save	keyboard

After synchronisation

⟨ {login},
{search, filter},
{open},
{save} ⟩

The time window defines which events are simultaneous.

Similarity between itemsets

Exact match

```
{search, filter} = {search, filter}
```

Partial match

$$J(A,B) = |A \cap B| \; / \; |A \cup B|$$

A	B	Jaccard similarity
{search, filter}	{search}	1 / 2
{open, save}	{open, print}	1 / 3
{login}	{logout}	0

The local similarity must reflect the interaction question.

Aligning sequences of itemsets

Routine A

```
⟨{login}, {search},       {search, filter}, {save}⟩
```

Routine B

```
⟨{login}, {search}, {select}, {search, filter}, {save}⟩
```

Possible alignment

```
A  {login}  {search}  { gap  }  {search, filter}  {save}
B  {login}  {search}  {select}  {search, filter}  {save}
```

Scoring choice	Role
Itemset similarity	Scores local matches
Gap penalty	Controls tolerated extra steps

Activity: specify an alignment model

```
A  ⟨{home}, {door}, {kitchen, light}, {coffee}⟩

B  ⟨{home}, {door}, {hall}, {kitchen, light}, {coffee}⟩

C  ⟨{home}, {door}, {kitchen}, {coffee, music}⟩
```

Using the concepts introduced on the previous slides, propose:

- What are the input objects?
- How should two itemsets be compared?
- How should an extra or missing step be penalised?
- What should the output contain?
- How would these choices affect the comparison of A with B and A with C?

Correction: a defensible model

Element	Possible choice
Input	Two sequences of (synchronised) itemsets
Local score	Jaccard similarity between itemsets
Gap cost	A fixed penalty for an extra or missing step
Output	The best alignment and a normalised similarity score

A versus B:
one gap corresponds to {hall}

A versus C:
J({kitchen, light}, {kitchen}) = 1/2
J({coffee}, {coffee, music}) = 1/2

Deciding whether two traces describe the same routine requires an additional threshold or clustering step

Evaluation: choose the correct level

Claim	Evaluation unit	Useful evidence
Detect routine occurrences	Occurrence	Precision, recall, F1
Recover routine types	Pattern	Coverage and expert matching
Generalise across people	User	Results on unseen users

- Ground truth must describe the same unit as the output.
- For unbalanced classes, report precision, recall and F1 for the target class or report class-specific scores and macro-F1
- Computation time matters when traces grow.

Interaction data need grouped splits

Random event split

Events from the same user or session appear in both training and test data. The measured score may reflect memorisation.

Grouped split

Keep complete users, sessions or time periods together. The split must match the intended deployment.

Deployment claim	Recommended split
Future sessions from known users	Chronological split by session
Behaviour of new users	User-disjoint split
Future system usage	Temporal holdout

Never split individual events from the same session across training and test data.

Frequent sequential patterns

Repeated behaviour while preserving order

An unordered set loses behavioural structure

Session 1

⟨login, search, open, save⟩

Session 2

⟨login, open, search, open, save⟩

If we retain only the actions that occurred (regardless of how many times)

{login, search, open, save}

{login, search, open, save}

What information has been lost?

Order	Repetitions	Adjacency and gaps	Duration
search before and after open	open occurs twice	intermediate actions disappear	lost without timestamps

Sets describe co-occurrence. Sequences describe ordered behaviour.

Sequential patterns and trace support

```
Trace S = ⟨login, search, open, save⟩
Pattern P = ⟨search, save⟩

P occurs in S because its events appear in the same order (with a little gap).
```

Trace support

```
support(P) = number of traces containing P / number of traces
```

A pattern repeated 20 times in one trace still contributes one trace to this support.

Activity: count trace support

```
S1  ⟨login, search, open, save⟩
S2  ⟨login, search, open, save⟩
S3  ⟨login, open, back, open, save⟩
S4  ⟨login, search, open, logout⟩
```

Pattern	Trace support	Frequent if minsup = 3?
⟨login, open⟩	?	?
⟨search, open⟩	?	?
⟨open, search⟩	?	?
⟨open, save⟩	?	?

Correction: trace support

```
S1  ⟨login, search, open, save⟩
S2  ⟨login, search, open, save⟩
S3  ⟨login, open, back, open, save⟩
S4  ⟨login, search, open, logout⟩
```

Pattern	Supporting traces	Support	Frequent?
⟨login, open⟩	S1, S2, S3, S4	4 / 4	Yes
⟨search, open⟩	S1, S2, S4	3 / 4	Yes
⟨open, search⟩	None	0 / 4	No
⟨open, save⟩	S1, S2, S3	3 / 4	Yes

Frequent individual events do not imply a frequent ordered pattern.

PrefixSpan: projected pattern growth

Step	Operation
1	Find frequent one-event prefixes.
2	For each prefix, retain the suffix that follows it in every trace.
3	Find frequent extensions in this projected database.
4	Extend the prefix and repeat recursively.

PrefixSpan searches projected databases instead of generating every candidate sequence.

Note: both Suffix trees and PrefixSpan exploit prefixes and suffixes to avoid repeatedly examining the complete data. However, suffix trees index contiguous substrings, whereas PrefixSpan recursively explores frequent subsequences through projected databases.

Example: a projected database

Prefix ⟨login, open⟩

```
S1  ⟨login, search, open, save⟩
S2  ⟨login, search, open, save⟩
S3  ⟨login, open, back, open, save⟩
S4  ⟨login, search, open, logout⟩
```

Projected suffixes

Trace	Suffix after the prefix
S1	⟨save⟩
S2	⟨save⟩
S3	⟨back, open, save⟩
S4	⟨logout⟩

save occurs in 3 projected suffixes
Frequent extension: ⟨login, open, save⟩

Note: at this stage, gaps between matched events are unrestricted

Constraints define meaningful patterns

Constraint	Question it answers
Minimum support	How widespread must the pattern be?
Maximum gap	How many intermediate events are acceptable?
Maximum duration	How much elapsed time is acceptable?
Maximum length	How complex may a reported pattern become?
Support unit	Sessions, users or episodes?

⟨open menu, select item⟩ may mean something different after 2 seconds and after 3 days.

Frequent patterns still need interpretation

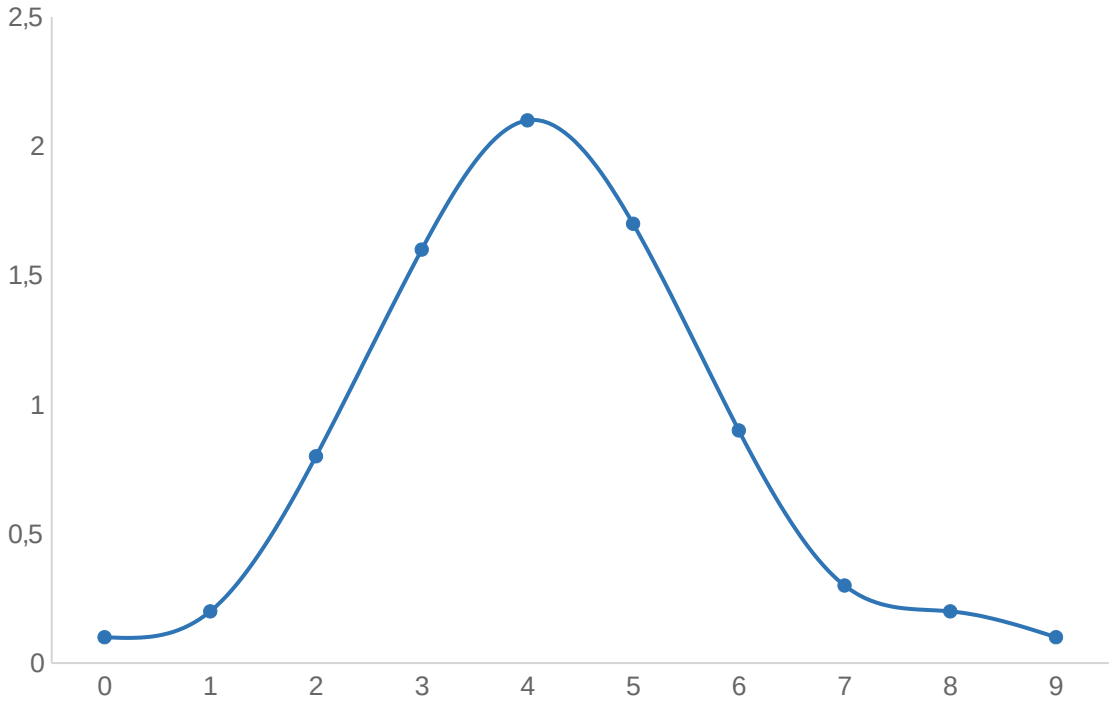
Pattern	Support	Possible interpretation
⟨login⟩	100%	Frequent but trivial
⟨open, save⟩	75%	Potentially useful workflow
⟨error, retry, abandon⟩	12%	Rare but important failure path

- Low thresholds can produce too many redundant patterns.
- Closed or maximal patterns reduce the number of outputs.
- A frequent pattern describes association, not causality.

Continuous and multivariate signals

Raw samples need an analysable representation

A continuous phenomenon becomes sampled data



- Choose a sampling rate.
- Handle noise and missing values.
- Resample when streams use different clocks.
- Preserve the temporal scale required by the task.

Sampling is already a modelling choice.

Discretisation changes the question

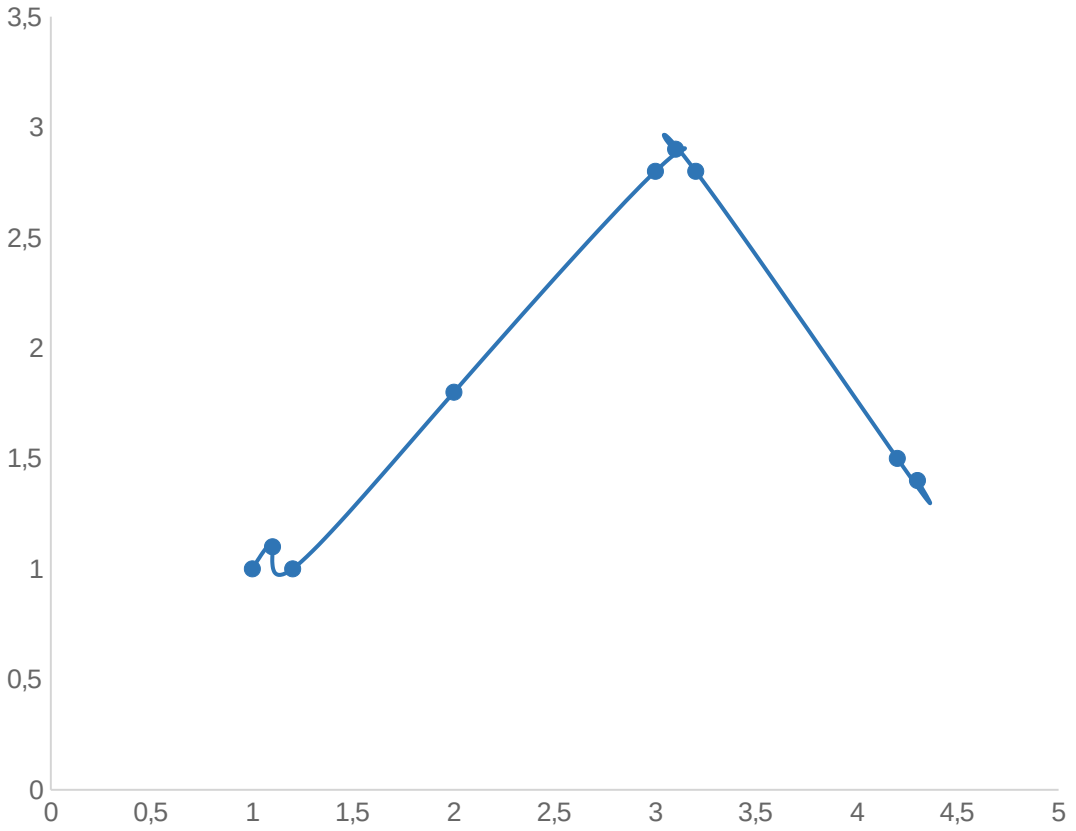
Time	Pointer speed	Symbol
t1	0.1	still
t2	0.4	slow
t3	1.3	fast
t4	1.7	fast
t5	0.3	slow

$$\langle \text{still}, \text{slow}, \text{fast}, \text{fast}, \text{slow} \rangle$$

- Fixed thresholds use domain knowledge.
- Clustering can learn data-driven states.
- Personal thresholds account for user differences.

Different discretisations produce different patterns.

GPS is a two-dimensional signal



Raw representation

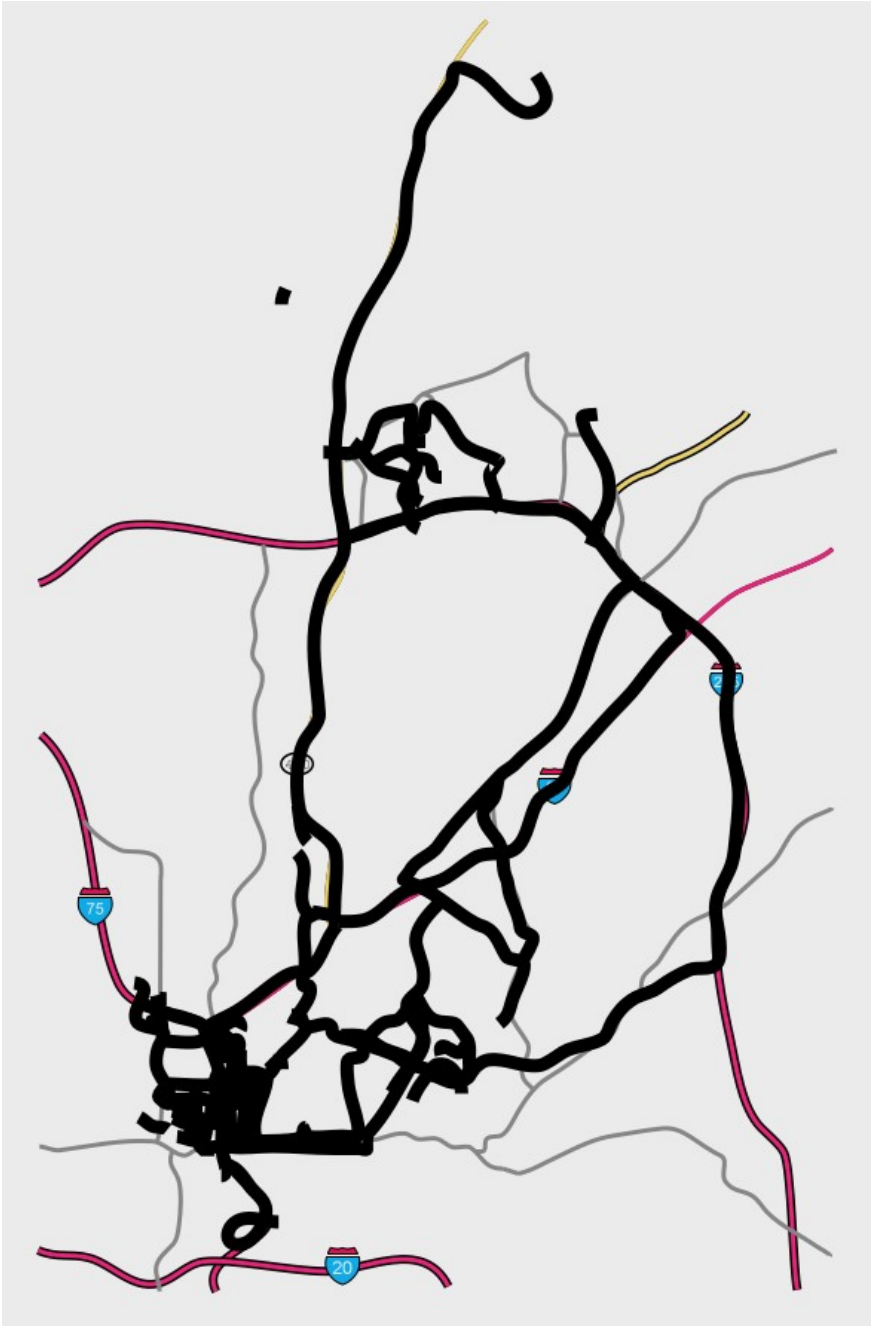
$\langle (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n) \rangle$

Symbolic representation

⟨Home, Campus, Shop, Home⟩

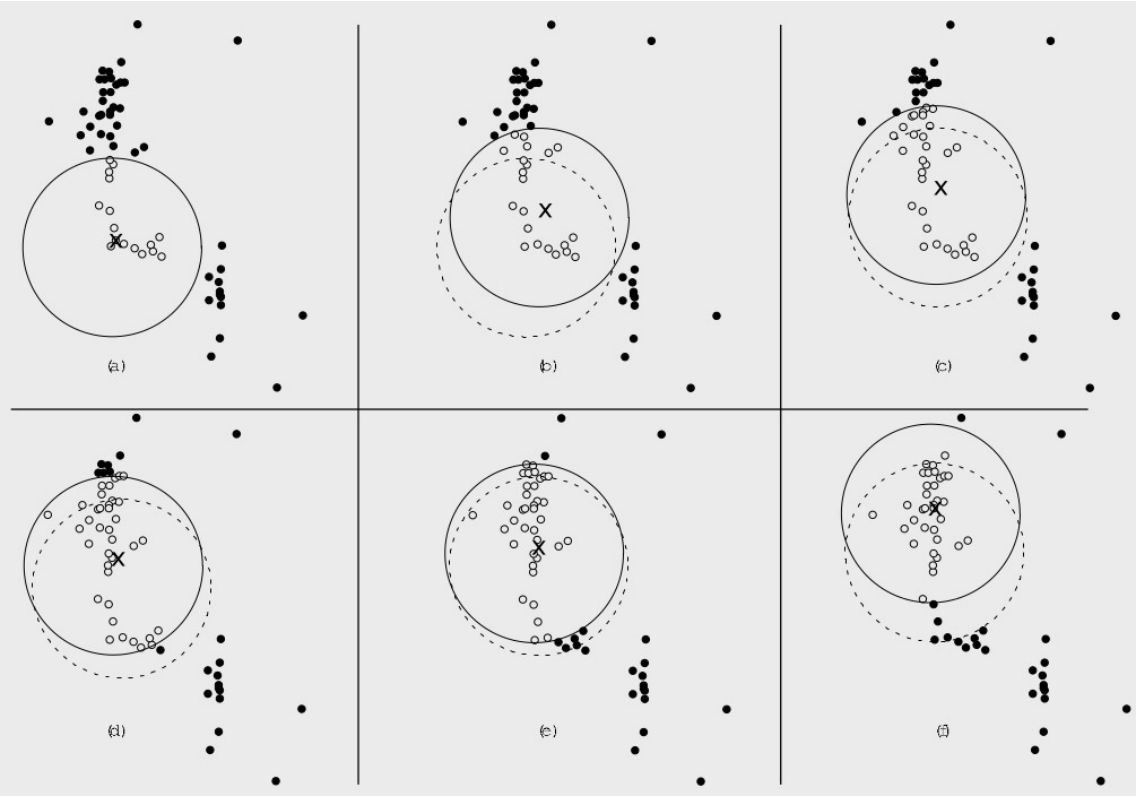
Place extraction requires a spatial radius and a dwell-time threshold.

Example: GPS signal

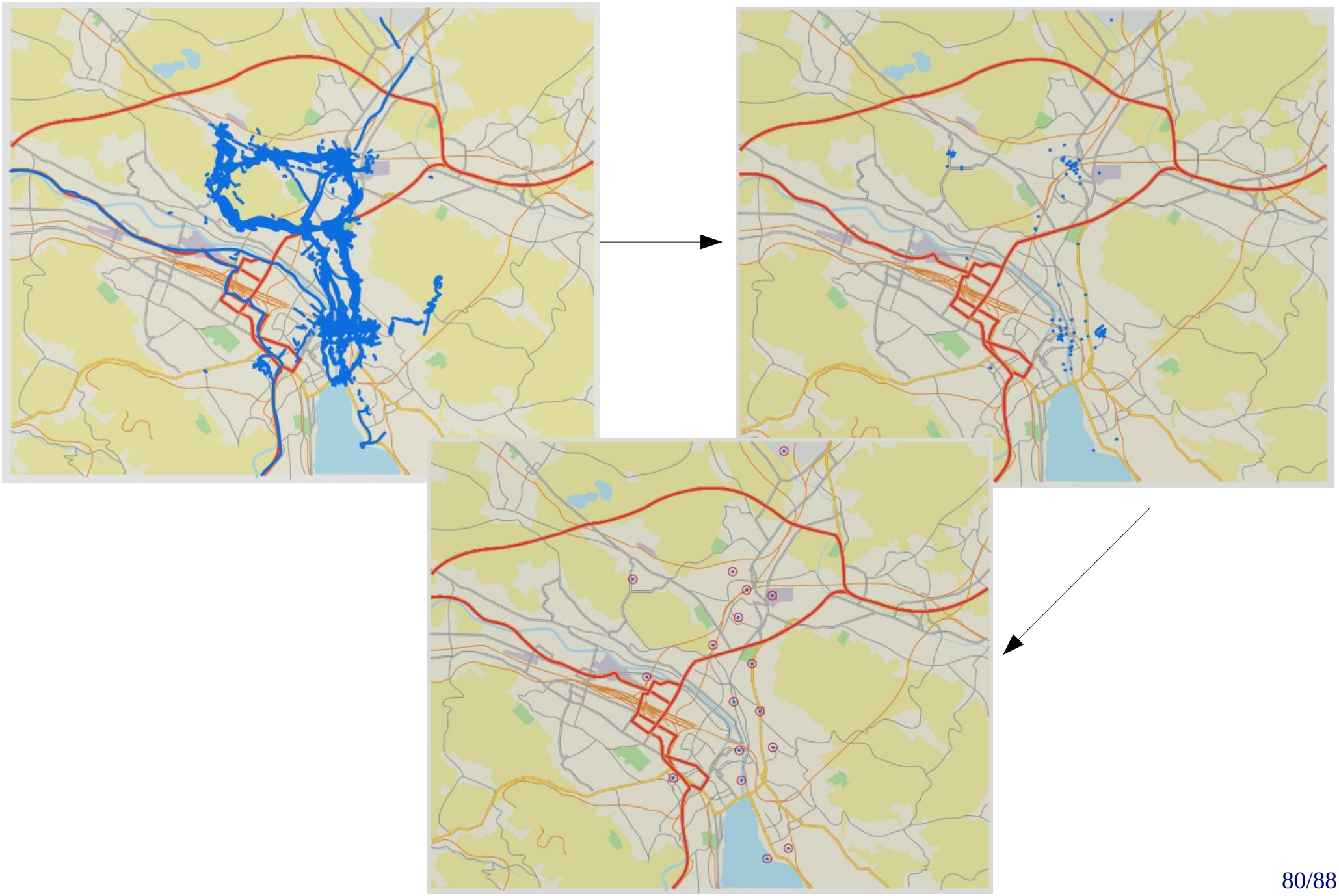


Daniel Ashbrook & Thad Starner : « Using GPS to learn significant locations and predict movement across multiple users », Personal and Ubiquitous Computing, volume 7, number 5, pp. 275-286, Springer, 2003.

- Finding **significant places / positions** (time dependence: time threshold)
- Clustering places into **locations / keypoints** (spacial dependence: cluster radius)



Example: GPS signal



Multiple sensors require synchronisation

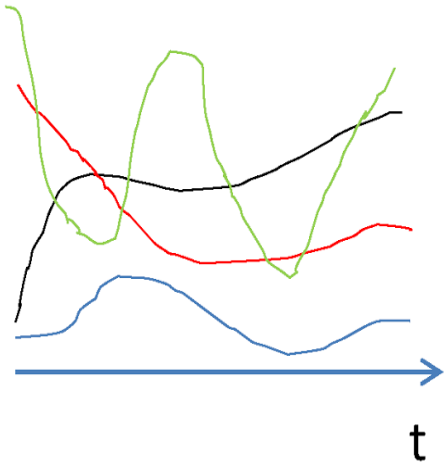
Stream	Typical timing	Example observation
Gaze	60 Hz	fixation coordinates
Pointer	event based	click or movement
Heart rate	1 Hz	beats per minute
Speech	interval based	utterance segment

Representation choice	Consequence
Common time grid	Direct comparison, possible interpolation
Event-centred windows	Preserves interaction episodes
Separate models	Combines results after independent processing

(Multiple) Continuous signals

$C = C(t)$

$\mathbb{R} \rightarrow \mathbb{R}^n$



discretisation

$C = (c_1(t_1), c_2(t_2), \ldots, c_n(t_n))$

$\mathbb{N}^n \rightarrow \mathbb{N}^n$

1.1	0.3	0.2	0.8
0.7		0.2	
0.4	0.5	0.3	
0.4	0.6	0.4	
0.6		0.5	0.5
0.8	0.3	0.2	
0.5	0.6	0.2	

t

synchronisation and classification

$C = (c_1, c_2, \ldots, c_n)(t)$

$\mathbb{N} \rightarrow \mathbb{N}^n$

A_1	A_2	A_3	A_4
B_1	A_2	A_3	A_4
C_1	B_2	A_3	A_4
C_1	B_2	B_3	A_4
B_1	B_2	B_3	B_4
B_1	A_2	A_3	B_4
C_1	B_2	A_3	B_4

t

Activity: formalise before choosing a method

Scenario

A learning platform records clicks, hint requests, submissions and pointer speed. We want to study recurrent help-seeking behaviour.

Field	Your formalisation
Input	
Representation	
Target output	
Problem class and method	
Evaluation and split	

Correction: two valid problem formulations

Discover recurrent behaviour

Element	Choice
Representation	Sequence of event itemsets and speed states
Output	Frequent sequential patterns
Method	PrefixSpan with temporal constraints
Evaluation	Support on held-out users

Predict a request for help

Element	Choice
Representation	Features from a fixed time window
Output	Probability of a future hint request
Method	Supervised classification
Evaluation	User-disjoint precision and recall

The target output determines the problem class.

CM2 synthesis

Representation	Structure preserved	Possible methods
Sequence of itemsets	Order and simultaneity	Alignment, PrefixSpan
Continuous signal	Amplitude and time	Time-series analysis, clustering
Symbolic sequence	Discrete states and order	PrefixSpan, HMM

Interaction phenomenon
Observable traces
Data representation
Target output
Problem class
Suitable method
Evaluation

Representation is part of the model.

Additional material

Reference slides not required during the 90-minute lecture

Apriori and PrefixSpan

Property	Apriori	PrefixSpan
Input	Unordered transactions	Ordered sequences
Output	Frequent itemsets	Frequent sequential patterns
Search	Candidate generation and pruning	Projected databases
Preserves order	No	Yes

Shared principle: if a pattern is not frequent, none of its extensions can be frequent.

References

• Pei, J. et al. (2001). PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth.

Proceedings of ICDM.

• Pei, J. et al. (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach.

IEEE Transactions on Knowledge and Data Engineering, 16(11).

• Agrawal, R. and Srikant, R. (1995). Mining Sequential Patterns.

Proceedings of ICDE.

• Ashbrook, D. and Starner, T. (2003). Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users.

Personal and Ubiquitous Computing, 7(5).