POLYTECH NANTES

M2 Data Science

Text and sequential pattern mining

Julien BLANCHARD

novembre 2017

Outline

- Introduction
- Episode Mining
- Sequential Pattern Mining
- Text Mining
- Process Mining

Outline

- Introduction
  - Patterns
  - Patterns in sequences
- Episode Mining
- Sequential Pattern Mining
- Text Mining
- Process Mining

What is a pattern?

- A **pattern** is a non-null finite series of **symbols** which appears frequently in data.
  Example: "blue-eyed and blonde-haired" in a group of persons.

- The nature of symbols highly depends on the **type of data**, since a symbol is a subset of data:
  - tabular data
  - transactional data
  - sequential data (temporal sequence, DNA sequence, text…)
  - hierarchical data (trees)
  - graph data
  - …

- The basic characteristic of a pattern is its **frequency**. Computing frequency requires **counting the occurrences** of the pattern in data.

"Data mining is the art of counting"
Prof. Heikki Mannila

What is a pattern?

Example of tabular dataset

Age	Sexe	Département	Revenus
[40;50]	H	44	15 à 30 k€
[20;30]	F	49	< 15 k€
[20;30]	F	44	15 à 30 k€
[50;60]	H	53	30 à 45 k€
[30;40]	H	22	< 15 k€
[50;60]	F	35	45 à 60 k€
[60;70]	H	35	15 à 30 k€

What is a pattern?

Example of transactional dataset (e.g. market basket dataset)

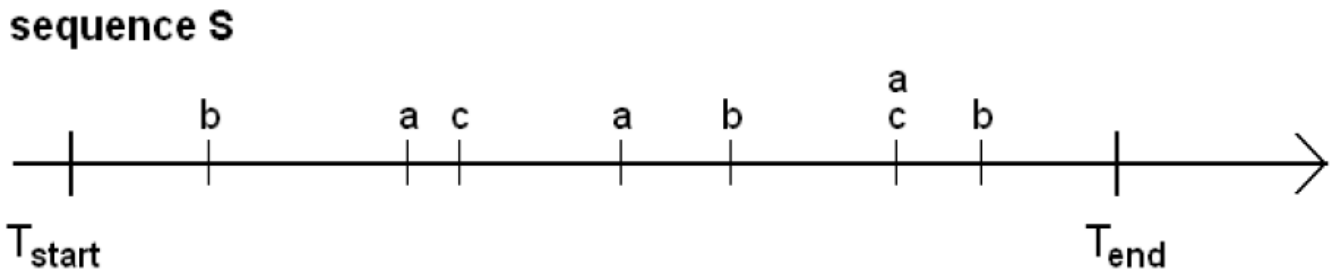
N° transaction	Huîtres	Muscadet	Citron	Foie gras	Vin
102155	1			1	
102156		1			
102157			1		1
102158	1	1	1		
102159			1	1	1
102160	1				
102161		1			

N° transaction	Item
102155	huîtres
102155	foie_gras
102156	muscadet
102157	citron
102157	vin
102158	huîtres
…	…

What is a pattern?

Example of sequential dataset:

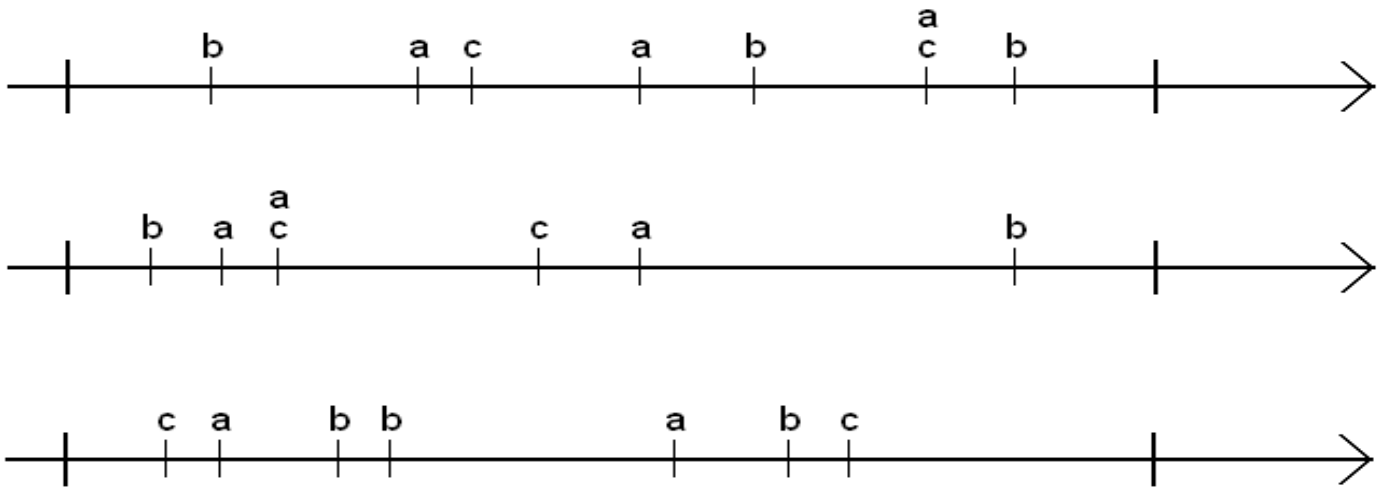
one long single sequence of events



What is a pattern?

Example of sequential dataset:

several sequences of events



What is a pattern?

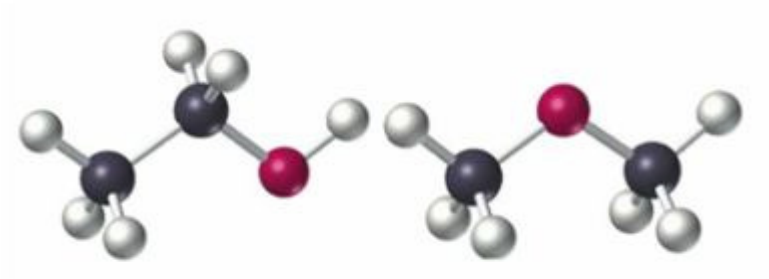
Example of graph dataset:

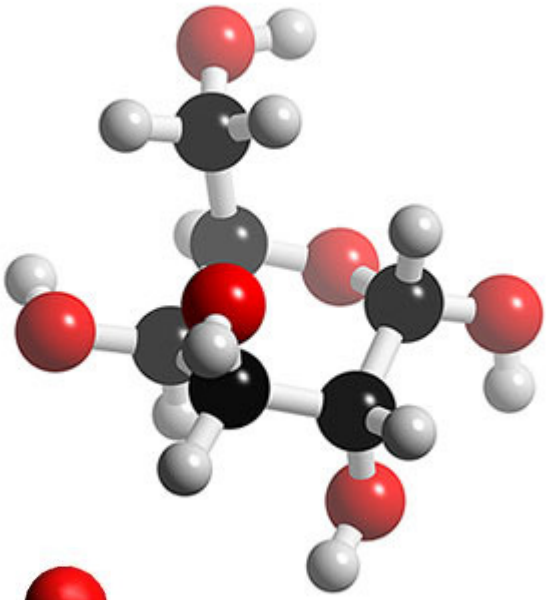
one large single graph (e.g. a social network)

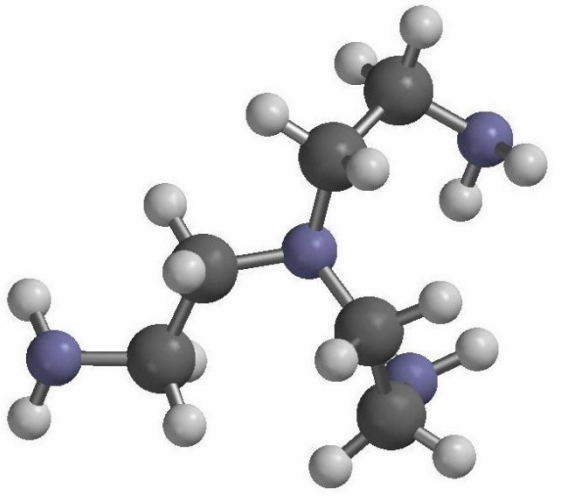
What is a pattern?

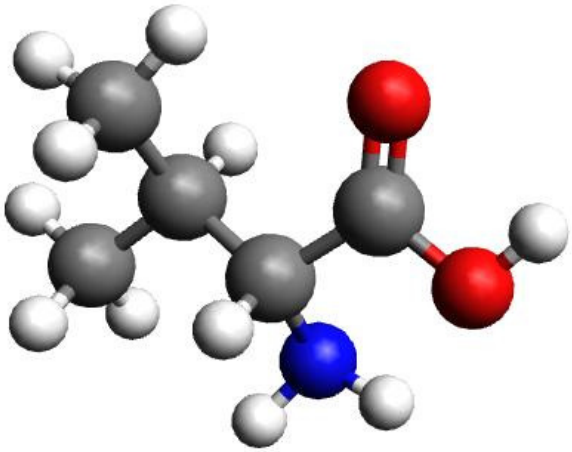
Example of graph dataset:

several graphs (e.g. molecules)









Why use patterns?

- Because patterns are very **intelligible** results that use the same "vocabulary" as data.

- Because a pattern allows focusing on a precise behavior concerning a precise group of individuals (**nugget**).

- Because combinatorial pattern mining algorithms allow discovering interesting patterns **that no one would have think they exist in the dataset**.

- Issue : **the profusion** of generated patterns!

From patterns to rules

- **Rules** are a special kind of patterns:

a pattern denoted $\alpha \rightarrow \beta$
with a left-hand side α and a right-hand side β

- Assessing rules requires **association measures**.

AJOUTER : Association Rules = Rules extracted from a database by an **association rule mining algorithm** !
(combinatorial algorithm)

Outline

- Introduction
  - Patterns
  - Patterns in sequences
- Episode Mining
- Sequential Pattern Mining
- Text Mining
- Process Mining

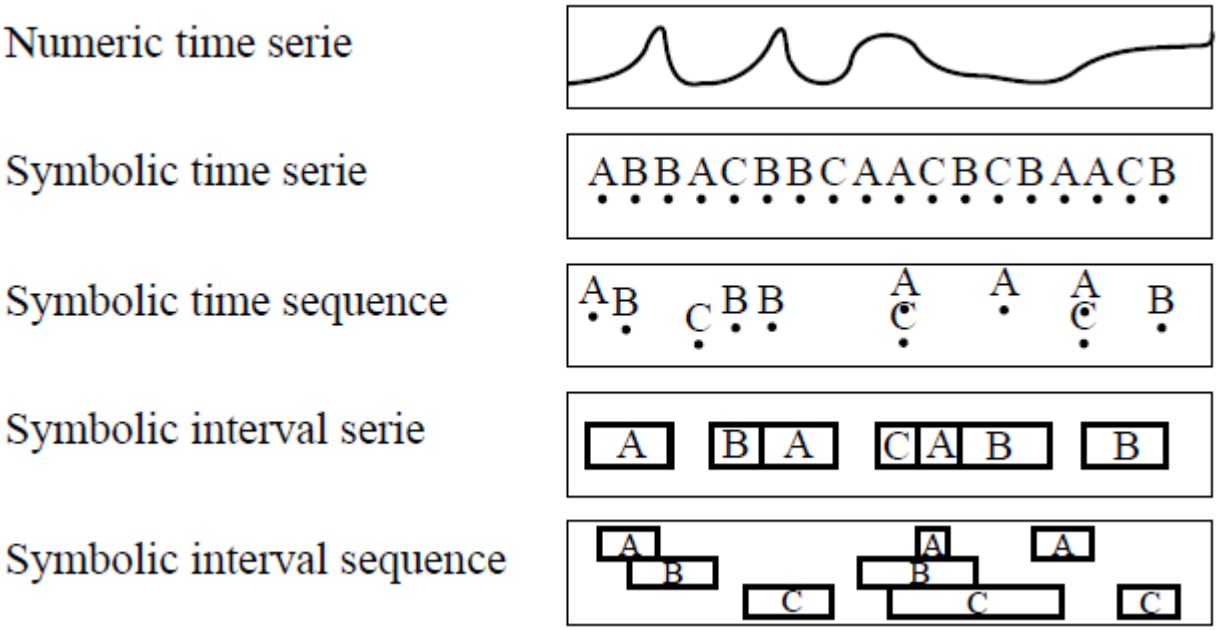
Patterns in sequences

- The general name is "**sequential pattern**", but there are many kinds of sequential patterns:
  - sequential patterns
  - time-interval sequential patterns
  - episodes
  - chronicles
  - temporal rules
  - cyclic association rules
  - …
- Even if you choose a precise type of patterns (e.g. episodes), there is no standard method to extract the patterns
  - → **unstandardized task** (unlike itemset mining or association rule mining) [Mooney and Roddick – 2013]
  - → the results (generated patterns) depend on the mining algorithm

Sequence characteristics

We are interested in sequences of symbolic events

→ no (numeric) time series!



[Guillaume-Bert – 2013]

Sequence characteristics

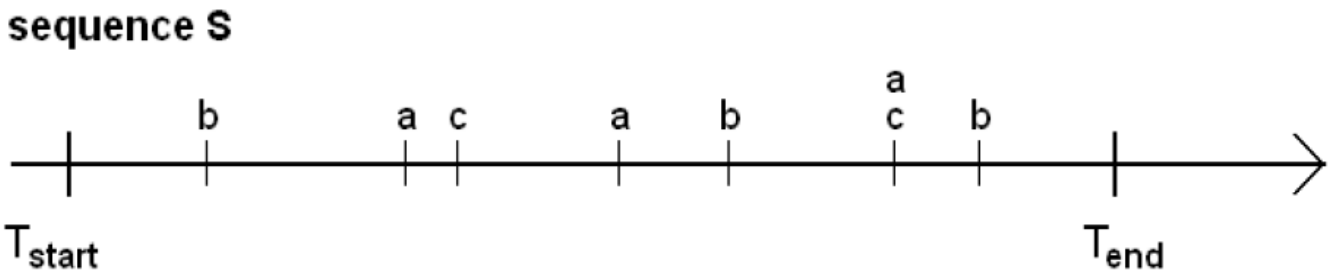
- A single sequence or several sequences?
- Which length?
- Continuous or discrete sequence?
- Instantaneous events? With a fixed/variable duration?
  Two events can appear at the same time?
- How many event types? (alphabet)

➔ Numerous types of sequences

➔ Numerous types of patterns

➔ Numerous algorithms

Pattern characteristics



Example of pattern: (a, b, c) "a then b then c"

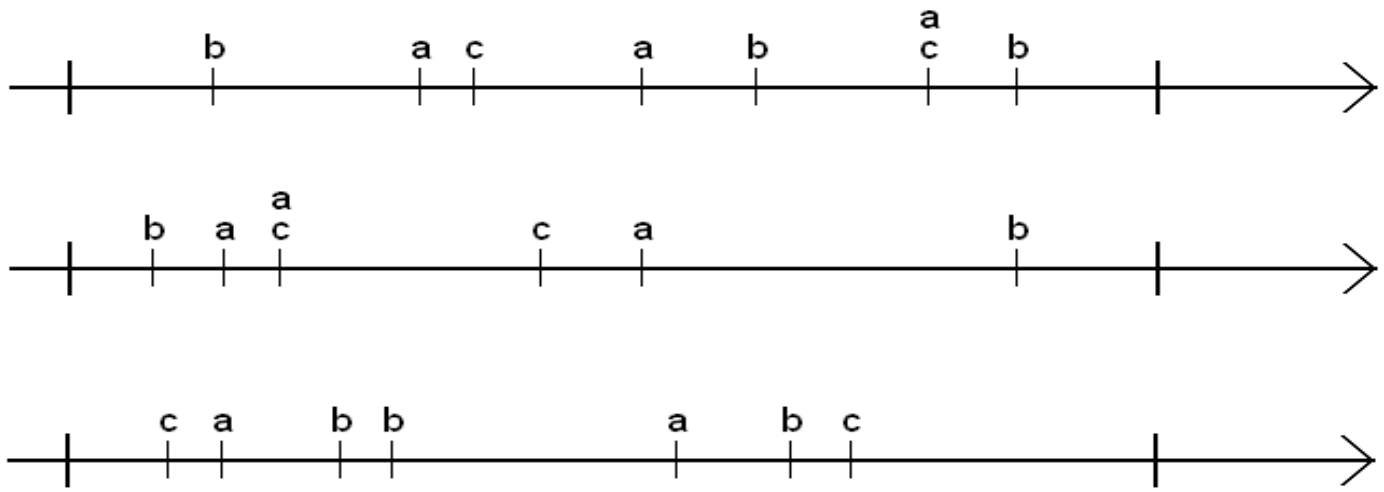
- Time constraint:
  - global or inter-events
  - min or max
- Ordered or unordered patterns (a, b, c) or {a, b, c}
- Mix? ({({a,b},c),(c,d)},e)

Two reference approaches

- *Frequent episode mining* in a long sequence of events
  [Mannila, Toivonen and Verkamo – 1995]

- *Frequent sequentiel pattern mining* in a set of event sequences
  [Agrawal and Srikant – 1995]

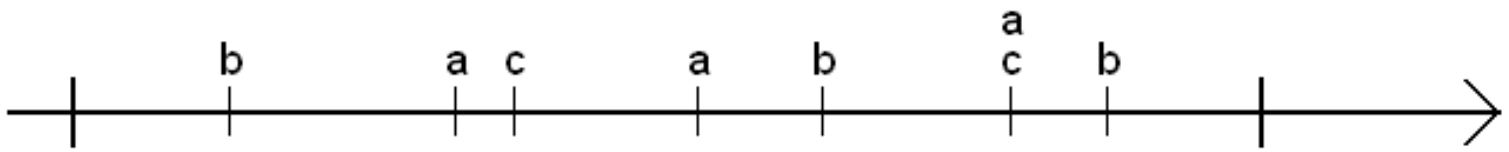
Sequential pattern mining



- Search for patterns which are repeated in the different sequences
- The frequency (support) of a pattern is measured in this way:

nb sequences with the pattern / total nb of sequences

Episode mining



- Search for patterns which are repeated in the sequence
- Several solutions are possible to measure frequency…

Outline

- Introduction
- Episode Mining
  - Counting strategies
  - Mining Algorithm
  - Properties and measures
- Sequential Pattern Mining
- Text Mining
- Process Mining

Episode counting strategies

- counting windows
- counting minimal windows (called "minimal occurrences")
- counting subsequences
- counting events with a stack
- counting events with a queue
- counting events with an anchor
- …

→ cf. exercise

[Mooney and Roddick – 2013, Gan and Dai – 2010]

Outline

- Introduction
- Episode Mining
  - Counting strategies
  - Mining algorithms
  - Properties and measures
- Sequential Pattern Mining
- Text Mining
- Process Mining

Sequence mining algorithms

- Three broad classes of algorithms:
  - Apriori-based algorithms using a horizontal database format
    - sequential patterns : GSP, PSP, SPIRIT
    - episodes: WINEPI, MINEPI
  - Apriori-based algorithms using a vertical database format
    - sequential patterns: SPADE, SPAM, LAPIN
  - Pattern-growth algorithms
    - sequential patterns: FreeSpan, PrefixSpan
    - episodes: PROWL

[Mooney and Roddick – 2013, Pei el al. – 2002]

WINEPI

The algorithm follows a traditional level-wise (breadth-first) search starting with the general episodes (one event). At each subsequent level the algorithm first computes a collection of candidate episodes, checks their frequency against *min_freq* and, if greater, the episode is added to a list of frequent episodes. This cycle continues until there are no more candidates generated or no candidates meet the minimum frequency. This is a typical Apriori-like algorithm under which the downward closure principal holds; if α is frequent then all subepisodes $\beta \preceq \alpha$ are frequent.

Algorithm

Input: A set E of event types, an event sequence $\mathbf{s}$ over E, a set $\mathcal{E}$ of episodes, a window width *win*, and a frequency threshold *min_fr*

Output: The collection $\mathcal{F}(\mathbf{s}, win, min\_fr)$ of frequent episodes.

Method:

1. $\mathcal{C}_1 := \{\alpha \in \mathcal{E} \mid |\alpha| = 1\}$;
2. $l := 1$;
3. *while* $\mathcal{C}_l \neq \emptyset$ *do*
4. /* Database pass: */
5. compute $\mathcal{F}_l := \{\alpha \in \mathcal{C}_l \mid fr(\alpha, \mathbf{s}, win) \geq min\_fr\}$;
6. $l := l + 1$;
7. /* Candidate generation: */
8. compute $\mathcal{C}_l := \{\alpha \in \mathcal{E} \mid |\alpha| = l$ and for all $\beta \in \mathcal{E}$ such that $\beta \prec \alpha$ and
9. $|\beta| < l$ we have $\beta \in \mathcal{F}_{|\beta|}\}$;
10. *for* all l *do* output $\mathcal{F}_l$;

Database scan in WINEPI for serial episodes

Episodes are recognized using **state automata** that accept the candidate episodes and ignore all other input.

- There is an automaton for each episode *e*, and there can be several instances of each automaton at the same time.
- A new instance of the automaton is initialized for each episode every time the first event appears in the window.
  The automaton instance is removed when this same event leaves the window.
- When an automaton reaches its accepting state, indicating that the episode *e* is entirely included in the window, and if there are no other automata for *e* in the accepting state already, we save the time of the window.
- When an automaton in the accepting state is removed, we increase the occurrence counter by the number of windows where *e* remained entirely in the window.
- **It is useless to have multiple automata of the same episode in the same state**, as they would only make the same transitions and produce the same information. It suffices to maintain the one that reached the common state last since it will be also removed last.