Knowledge Discovery and Data Mining 1
(Data Mining Algorithms 1)

Wintersemester 2019/20
1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   5.1 Introduction
   5.2 Process Model/Transition Systems
   5.3 Process Discovery
   5.4 Conformance Checking
   5.5 Additional Mining Tasks
Agenda

1. Introduction
2. Basics
3. Supervised Methods
4. Unsupervised Methods
5. Process Mining
   5.1 Introduction
   5.2 Process Model/Transition Systems
   5.3 Process Discovery
   5.4 Conformance Checking
   5.5 Additional Mining Tasks
Motivation

• Process models are generated either normative or descriptive
  
  • **Normative**: - invented by human
    - represent how a certain process is supposed to work
  
  • **Descriptive**: - created by process discovery algorithms based on log files
    - represent how a certain process is actually running
Process Discovery Algorithm „α-Miner“[1]

Idea: A simple algorithm to visualize processes

Input: Event log $L$ over activities $A$
Output: marked petri net / Workflow net

1. Detect log-based ordering relations from event Log $L$
2. Create Footprint Table
3. Execute the algorithm of the $\alpha$-Miner
4. Derive the WF-net

Process Discovery Algorithm „α-Miner“

Let \( L \) be an event log over activities \( A \), and let \( a, b \in A \).

1. *Detect log-based ordering relations from event Log \( L \)*

   i. „(direct) following“-relation \( a >_L b \)
      \[ \Leftrightarrow \exists \ text{trace } \sigma = \langle t_1, t_2, t_3, ..., t_{n-1} \rangle \text{ and } i \in \{1, 2, ..., n-2\} \]
      s. t. \( \sigma \in L \) and \( t_i = a \) and \( t_{i+1} = b \) and \( t_{i+1} = b \).

   ii. „potential parallelism“ \( a \parallel_L b \)
      \[ \Leftrightarrow a >_L b \text{ and } b >_L a \]

   iii. „sequential task“-relation \( a \rightarrow_L b \)
      \[ \Leftrightarrow a >_L b \text{ and } b \gg_L a \]

   iv. „not followed“-relation \( a \#_L b \)
      \[ \Leftrightarrow a \gg_L b \text{ and } b \gg_L a \]

\[ L = [(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4] \]

2. Create Footprint Table:
   i) Find the directly followed tupels

\[
\begin{array}{cccc}
 & a & b & c & d \\
 a &   &   &   &   \\
b &   &   &   &   \\
c &   &   &   &   \\
d &   &   &   &   \\
\end{array}
\]

\(>_L: \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\} \)
Process Discovery Algorithm „α-Miner“

Let $L$ be an event log over activities $A$, and let $a, b \in A$.

1. **Detect log-based ordering relations from event Log $L$**
   
   i. **(direct) following**-relation $a >_L b$
   
   $\Leftrightarrow \exists$ trace $\sigma = \langle t_1, t_2, t_3, ..., t_{n-1} \rangle$ and $i \in \{1, 2, ..., n-2\}$
   
   s. t. $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. **potential parallelism** $a \parallel_L b$

   $\Leftrightarrow a >_L b$ and $b >_L a$

   iii. **sequential task**-relation $a \rightarrow_L b$

   $\Leftrightarrow a >_L b$ and $b \nRightarrow_L a$

   iv. **not followed**-relation $a \#_L b$

   $\Leftrightarrow a \nRightarrow_L b$ and $b \nRightarrow_L a$

$L = [(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4]$

2. Create Footprint Table:
   ii) Find the potential parallel tuple and mark them in the table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$>_L$: \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\}

$\parallel_L$: \{(c, d), (d, c)\}
Process Discovery Algorithm „α-Miner”

Let $L$ be an event log over activities $A$, and let $a, b \in A$.

1. Detect log-based ordering relations from event Log $L$

   i. „(direct) following“-relation $a >_L b$
      \[ \iff \exists \text{ trace } \sigma = \langle t_1, t_2, t_3, \ldots, t_{n-1} \rangle \text{ and } i \in \{1,2,\ldots,n-2\} \]
      s. t. $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$ and $t_{i+1} = b$.

   ii. „potential parallelism“ $a \parallel_L b$
       \[ \iff a >_L b \text{ and } b >_L a \]

   iii. „sequential task“-relation $a \rightarrow_L b$
        \[ \iff a >_L b \text{ and } b \not>_L a \]

   iv. „not followed“-relation $a \#_L b$
        \[ \iff a \not>_L b \text{ and } b \not>_L a \]

2. Create Footprint Table:
   iii) Find the sequential task tupels and mark them in the table

\[
L = [(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4]
\]

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{c} & \text{d} \\
\hline
\text{a} & \rightarrow_L & \rightarrow_L & \\
\text{b} & \rightarrow_L & \rightarrow_L & \\
\text{c} & & \parallel_L & \\
\text{d} & & \parallel_L & \\
\end{array}
\]

$>_L: \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\}$

$\parallel_L: \{(c,d), (d,c)\}$

$\rightarrow_L: \{(a, c), (a, d), (b, c), (b, d)\}$
Process Discovery Algorithm „α-Miner“

Let \( L \) be an event log over activities \( A \), and let \( a, b \in A \).

1. Detect log-based ordering relations from event Log \( L \)
   i. “(direct) following”-relation \( a >_L b \)
      \[ \Leftrightarrow \exists \ trace \ \sigma = \langle t_1, t_2, t_3, \ldots, t_{n-1} \rangle \text{ and } i \in \{1, 2, \ldots, n-2\} \]
      s. t. \( \sigma \in L \) and \( t_i = a \) and \( t_{i+1} = b \) and \( t_{i+1} = b \).

   ii. “potential parallelism” \( a \parallel_L b \)
      \[ \Leftrightarrow a >_L b \text{ and } b >_L a \]

   iii. “sequential task”-relation \( a \rightarrow_L b \)
      \[ \Leftrightarrow a >_L b \text{ and } b \not>_L a \]

   iv. “not followed”-relation \( a \#_L b \)
      \[ \Leftrightarrow a \not>_L b \text{ and } b \not>_L a \]

\( L = [(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4] \)

2. Create Footprint Table:
   iv) Find the not followed tupels and mark them in the table

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{c} & \text{d} \\
\hline
\text{a} & \#_L & \#_L & \rightarrow_L & \rightarrow_L \\
\text{b} & \#_L & \#_L & \rightarrow_L & \rightarrow_L \\
\text{c} & & & \#_L & \parallel_L \\
\text{d} & & & \parallel_L & \#_L \\
\end{array}
\]

\( >_L : \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\} \)
\( \parallel_L : \{(c, d), (d, c)\} \)
\( \rightarrow_L : \{(a, c), (a, d), (b, c), (b, d)\} \)
\( \#_L : \{(a, a), (a, b), (b, a), (b, b), (c, c), (d, d)\} \)
Let \( L \) be an event log over activities \( A \), and let \( a, b \in A \).

1. **Detect log-based ordering relations from event Log \( L \)**

   i. **“(direct) following”-relation** \( a >_L b \)
      \[
      \iff \exists \text{ trace } \sigma = \langle t_1, t_2, t_3, \ldots, t_{n-1} \rangle \text{ and } i \in \{1,2,\ldots,n-2\} \text{ s. t. } \sigma \in L \text{ and } t_i = a \text{ and } t_{i+1} = b \text{ and } t_{i+1} = b.
      \]

   ii. **“potential parallelism”** \( a \parallel_L b \)
      \[
      \iff a >_L b \text{ and } b >_L a
      \]

   iii. **“sequential task”-relation** \( a \rightarrow_L b \)
      \[
      \iff a >_L b \text{ and } b \nRightarrow_L a
      \]

   iv. **“not followed”-relation** \( a \#_L b \)
      \[
      \iff a \nRightarrow_L b \text{ and } b \nRightarrow_L a
      \]

\( L = \{(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4\} \)

2. Create Footprint Table:
   v. The remaining tuples represent a “directly before” relation, marked as \( \leftarrow_L \) and mark them in the table:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>#_L</td>
<td>#_L</td>
<td>→_L</td>
<td>→_L</td>
</tr>
<tr>
<td>b</td>
<td>#_L</td>
<td>#_L</td>
<td>→_L</td>
<td>→_L</td>
</tr>
<tr>
<td>c</td>
<td>←_L</td>
<td>←_L</td>
<td>#_L</td>
<td>→_L</td>
</tr>
<tr>
<td>d</td>
<td>←_L</td>
<td>←_L</td>
<td>#_L</td>
<td>→_L</td>
</tr>
</tbody>
</table>

\( >_L: \{(a, c), (a, d), (b, c), (b, d), (c, d), (d, c)\} \)
\( \parallel_L: \{(c,d), (d,c)\} \)
\( \rightarrow_L: \{(a, c), (a, d), (b, c), (b, d)\} \)
\( \#_L: \{(a,a), (a,b), (b,a), (b,b), (c,c), (d,d)\} \)
Process Discovery Algorithm „α-Miner“

3. Execute the algorithm of the α-Miner
   
i) All activities that start any trace yield the set of **starting activities**, collected in $T_{in}$.
   
ii) All activities that end any trace yield the set of **output activities**, $T_{out}$.
   
…

4. Derive the WF-net:
   
• The set of **transitions** is equal to $A$, so each activity represents a transition
   
• A starting place is created and connected to each node in $T_{in}$.
   
• Also, a final place is created and each node in $T_{out}$ is connected to it.

$L = \{(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4\}$

$T_{in} = \{a, b\}$

$T_{out} = \{c, d\}$

```
L = [(a, c, d)^3, (a, d, c)^2, (b, c, d)^2, (b, d, c)^4]

T_{in} = \{a, b\}

T_{out} = \{c, d\}
```
3. Execute the algorithm of the $\alpha$-Miner …

   iii) Determine all pairs of sets $A$ and $B$, such that
   
   • $\forall a_1, a_2 \in A: a_1 \# a_2$
   • $\forall b_1, b_2 \in B: b_1 \# b_2$
   • $\forall a_1 \in A, \forall b_1 \in B: a_1 \rightarrow b_1$
   
   • Select only the “maximal pairs”:
     e.g. ($\{a\}, \{c\}$), ($\{a\}, \{d\}$), ($\{a\}, \{c, d\}$) $\Rightarrow$ ($\{a\}, \{c, d\}$)

4. A place is added in between $A$ and $B$ and connected accordingly

   e.g. $A = \{a\}, B = \{b, e\}$

Heuristics-Miner is our first algorithm to capture concurrent process behavior.

valid set of „maximal pairs“:
($\{a\}, \{c, d\}$), ($\{b\}, \{c, d\}$)

**Idea:** $\alpha$-Miner has several flaws (1-loops, 2-loops, no weighting).

Heuristics-Miner uses dependency as the condition to connect activities.

**Input:** Event log $L$

**Output:** Causal net, *here we stop at the dependency graph*

---

Let $L$ be an event log over activities $A$, and let $a, b \in A$.

1. Create table displaying frequency of „directly follows“ relation $>_L$

$L = [(a, e)^5, (a, b, c, e)^{10}, (a, c, b, e)^{10}, (a, b, e)^{10}, (a, d, d, e)^2, (a, d, d, d, e)^1]$
2. Create a table showing the value of „dependency measures“ of all pairs of activities over $L$

$$|a \Rightarrow_L b| = \begin{cases} \frac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1}, & \text{if } a \neq b \\ \frac{|a > a|}{|a > a| + 1}, & \text{if } a = b \end{cases}$$

$$|a \Rightarrow_L b| \in ]-1,1[$$

$$|a \Rightarrow_L b| = 0\quad \text{, if } |a >_L b| = |b >_L a|$$

$$|a \Rightarrow_L b| \rightarrow 1\quad \text{, if } a \text{ follows almost always after } b$$

|\Rightarrow_L| a b c d e
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
2. Create a table showing the value of „dependency measures“ of all pairs of activities over $L$

$$|a \Rightarrow_L b| = \begin{cases} 
\frac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1}, & \text{if } a \neq b \\
\frac{|a > a|}{|a > a| + 1}, & \text{if } a = b 
\end{cases}$$

$$|a \Rightarrow_L b| \in [-1,1[$$

$$|a \Rightarrow_L b| = 0, \text{ if } |a >_L b| = |b >_L a|$$

$$|a \Rightarrow_L b| \rightarrow 1, \text{ if } a \text{ follows almost always after } b$$

Lower triangular matrix is the negative and transposed of the upper triangular matrix.

\[
\begin{array}{cccccc}
| \Rightarrow_L | & a & b & c & d & e \\
\hline
a & 0 & 0.92 & 0.92 & 0.93 & 0.83 \\
b & -0.92 & 0 & 0 & 0 & 0.92 \\
c & -0.92 & 0 & 0 & 0 & 0.92 \\
d & -0.93 & 0 & 0 & 0.80 & 0.93 \\
e & -0.83 & -0.92 & -0.92 & -0.93 & 0 \\
\end{array}
\]

\[
|a \Rightarrow_L b| = \frac{11 - 0}{11 + 0 + 1} = 0.92
\]

\[
|b \Rightarrow_L c| = \frac{10 - 10}{10 + 10 + 1} = 0
\]
3. i) Select **two thresholds** to reduce noise \((\tau_{\geq L})\) and infrequent traces \((\tau_{\Rightarrow L})\)

ii) Create the dependency graph DG:
- an arc between \(x\) and \(y\) is only included if 
  \[ |x <_L y| \geq \tau_{\geq L} \land |x \Rightarrow_L y| \geq \tau_{\Rightarrow L} \]

**Ex. 1:**
- Setting \(\tau_{\geq L} = 2\) and \(\tau_{\Rightarrow L} = 0.7\)
- yields to the following dependency graph:
Process Discovery Algorithm „Heuristics-Miner“

3. i) Select **two thresholds** to reduce noise ($\tau_{>L}$) and infrequent traces ($\tau_{\Rightarrow L}$)

ii) Create the dependency graph $DG$: an arc between $x$ and $y$ is only included if $|x <_L y| \geq \tau_{>L}$ and $|x \Rightarrow L y| \geq \tau_{\Rightarrow L}$

**Ex. 2:**
Setting $\tau_{>L} = 5$ and $\tau_{\Rightarrow L} = 0.9$
yields to the following dependency graph:
Process Discovery Algorithm „Heuristics-Miner“

4. Last step – *not in this lecture*: dependency graph → causal net
5. Process Discovery

- „Inductive-Miner (IM)” [3]:
  It uses the directly-follows graph that corresponds to the „direct follows“ relation ($\alpha$) used by the $\alpha$-Miner and creates a Process Tree $Q$.

- „Declare” [4]:
  It is a constrained based declarative approach.

---


1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods

5. Process Mining
   5.1 Introduction
   5.2 Process Model/Transition Systems
   5.3 Process Discovery
   5.4 Conformance Checking
   5.5 Additional Mining Tasks
Motivation

• Given an event log and a process model, decide for each case whether it conforms to the model or not. If not, give the issues.
Goal: Fraud detection

- Alteration of medical treatment, usually for higher compensations ("upcoding"). Cheap medication billed as costly medication. Medication is non-conform to the treatment plan, e.g. flu vaccination after broken leg.

- Duplicate execution of actions. Billing twice for same service or good

- Embezzlement, theft or misuse of company assets. Usage of company truck at suspicious times for private actions (evenings, vacation,...), or faked payments using complex and unusual cashflows.
Goal: Workflow improvements

- Root-cause detection
  Quality check failed for some products. Search for shared historic activities (e.g. same supplier, preprocessed by same employee or machine, similar environmental conditions).

- Standardization of deviations
  Customers are processed faster at a certain counter. How has the employee deviated the process? E.g. Families with children board first at the airport.

- Customer aggregation
  Some customers look for furniture in a popular shop. The order of furniture presentation influences their habits. Where to offer the small items like tealights? Which customer types map to which market traversal paths?
Automata Theory: Decide Language Membership

- Idea:
  - Put a token into the start position.
  - For each event, fire the transition with the same label in the Petri net.
  - If the Petri net accepts the sequence, the trace passed the conformance checking.
  - Otherwise, a rejected trace has zero fitness.

Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced: 0

(c)onsumed: 0
Petri Net Membership Test

Checking: <Avocado, Rice, Salmon, Nori, Eat>

Produced: 1
Consumed: 0
Checking: \(<\text{Avocado}, \text{Rice}, \text{Salmon}, \text{Nori}, \text{Eat}>\)

\(\text{(p)roduced : 4} \quad \text{(c)onsumed : 1}\)
Petri Net Membership Test

Checking: \(<\text{Avocado}, \text{Rice}, \text{Salmon}, \text{Nori}, \text{Eat}>\)

(produced : 5  consumed : 2)
Petri Net Membership Test

Checking: \(<\text{Avocado, Rice, Salmon, Nori, Eat}>\)

(p)roduced : 6  \hspace{1cm} (c)onsumed : 3
Petri Net Membership Test

Checking: \(<\text{Avocado, Rice, Salmon, Nori, Eat}>\)

(p)roduced: 7  (c)onsumed: 4
Petri Net Membership Test

Checking: <Avocado, Rice, Salmon, Nori, Eat>
(p)roduced: 8  (c)onsumed: 7
Petri Net Membership Test

Checking: <Avocado, Rice, Salmon, Nori, Eat>

(p)roduced : 9               (c)onsumed : 8
Checking:  <Avocado, Rice, Salmon, Nori, Eat>
(p)roduced :  9  (c)onsumed :  9
The fitness of a case with trace $\sigma$ on WF-net $M$ is defined as:

$$fitness(\sigma, M) = \frac{1}{2} \left( 1 - \frac{m}{c} \right) + \frac{1}{2} \left( 1 - \frac{r}{p} \right)$$

Considering the example:

Checking: $\sigma = \langle$Avocado, Rice, Salmon, Nori, Eat$\rangle$

(p)roduced: 9  (c)onsumed: 9

$$fitness(\sigma, M) = \frac{1}{2} \left( 1 - \frac{0}{9} \right) + \frac{1}{2} \left( 1 - \frac{0}{9} \right) = 1$$
Token Replay

- Problem with pure Automata approach:
  - We cannot decide between almost fit and critically deviating traces (binary classifier).
  - In practical applications we often need some flexibility to execute the processes.

- Modified Idea:
  - Put a token into the start position.
  - For each event, try to fire the corresponding transition in the net.
  - If not possible, create a virtual new token after the transition.
  - In the end, determine the fitness based on the tokens left in the model and the virtually added ones.

---

Token Replay Example

Checking: <Rice, Salmon, Wasabi>

(p)roduced : 1  (c)onsumed : 0
(m)issing : 0  (r)emaining : 0

5. Process Mining 5.4 Conformance Checking
5. Process Mining

5.4 Conformance Checking

Token Replay Example

Checking: <Rice, Salmon, Wasabi>

(p)roduced: 4  (c)onsumed: 1
(m)issing: 0  (r)emaining: 0
Token Replay Example

Checking: <Rice, Salmon, Wasabi>

(p)roduced: 5  (c)onsumed: 2
(m)issing: 0  (r)emaining: 0
Token Replay Example

Checking: <Rice, Salmon, Wasabi>

(p)roduced : 6   (c)onsumed : 3
(m)issing : 0   (r)emaining : 0

5. Process Mining

5.4 Conformance Checking
Token Replay Example

Checking:  <Rice, Salmon, Wasabi>

(p)roduced :   7   (c)onsumed :   4
(m)issing :    1   (r)emaining :  0
Token Replay Example

Checking:  <Rice, Salmon, Wasabi>
(p)roduced :  7  (c)onsumed :  4
(m)issing :  1  (r)emaining :  4
Token Replay Example

The fitness of a case with trace $\sigma$ on WF-net $M$ is defined as:

$$fitness(\sigma, M) = \frac{1}{2} \left( 1 - \frac{m}{c} \right) + \frac{1}{2} \left( 1 - \frac{r}{p} \right)$$

Considering the example:

Checking: $\sigma = \langle \text{Rice, Salmon, Wasabi} \rangle$

(p)roduced: 7  
(c)onsumed: 4

(m)issing: 1  
(r)emaining: 4

$$fitness(\sigma, M) = \frac{1}{2} \left( 1 - \frac{1}{4} \right) + \frac{1}{2} \left( 1 - \frac{4}{7} \right) = 0.375$$
Token Replay: Discussion

• Allows a continuous fitness score in the interval [0,1].

• Intuitive and easy to implement.

• For critical deviating behavior, model gets flooded with tokens. Earlier deviations mask later deviations.
  → all behavior afterwards gets accepted, fitness values too low

• Depending on a Petri net representation of the process.
Alignments²

• To overcome drawbacks of Token Replay, it might be better to map observed behavior on modelled behavior.

• Idea:
  • Consider all mappings between a model and a trace.
  • Simulate moves in the model and in the trace.
  • Optimize for most synchronous moves (fire transition $a$ and read $a$ in the trace in parallel).
  • Finally, compare the optimal alignment with the worst alignment possible to determine the fitness.

![](moves.png)

**moves in the log**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>&gt;&gt;</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
</table>

**moves in the model**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>f</th>
<th>&gt;&gt;</th>
<th>h</th>
</tr>
</thead>
</table>

>> is an asynchronous move

---

• Worst possible alignment for <Rice, Salmon, Wasabi>:

<table>
<thead>
<tr>
<th>Rice</th>
<th>Salmon</th>
<th>Eat</th>
<th>&gt;&gt;</th>
<th>&gt;&gt;</th>
<th>&gt;&gt;</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;</td>
<td>&gt;&gt;</td>
<td>&gt;&gt;</td>
<td>Rice</td>
<td>Avocado</td>
<td>Salmon</td>
<td>Nori sheet</td>
</tr>
</tbody>
</table>
Alignments

- Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

<table>
<thead>
<tr>
<th>Rice</th>
<th>&gt;&gt;</th>
<th>Salmon</th>
<th>Salmon</th>
<th>Wasabi</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td></td>
<td>Avocado</td>
<td>Salmon</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Wasabi</td>
<td>Eat</td>
</tr>
</tbody>
</table>
Alignments

• Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

<table>
<thead>
<tr>
<th></th>
<th>&gt;&gt;</th>
<th>Salmon</th>
<th>Salmon</th>
<th>Wasabi</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rice</td>
<td>Avocado</td>
<td>Salmon</td>
<td></td>
<td>Wasabi</td>
<td>Eat</td>
</tr>
</tbody>
</table>

• Optimal alignments do not require to be unique:

<table>
<thead>
<tr>
<th></th>
<th>&gt;&gt;</th>
<th>Salmon</th>
<th>Salmon</th>
<th>Wasabi</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rice</td>
<td>Avocado</td>
<td>&gt;&gt;</td>
<td>Salmon</td>
<td>Wasabi</td>
<td>Eat</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;</td>
<td>Rice</td>
<td>Salmon</td>
<td>Salmon</td>
<td>Wasabi</td>
</tr>
<tr>
<td>Avocado</td>
<td>Rice</td>
<td>&gt;&gt;</td>
<td>Salmon</td>
<td>Wasabi</td>
<td>Eat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Salmon</th>
<th>&gt;&gt;</th>
<th>Salmon</th>
<th>Wasabi</th>
<th>&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td></td>
<td>&gt;&gt;</td>
<td>Avocado</td>
<td></td>
<td>Eat</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;</td>
<td>Rice</td>
<td>Salmon</td>
<td>Salmon</td>
<td>Wasabi</td>
</tr>
<tr>
<td>Avocado</td>
<td>Rice</td>
<td>&gt;&gt;</td>
<td>Salmon</td>
<td></td>
<td>Wasabi</td>
</tr>
</tbody>
</table>

• However, the distance between log and model equal for all optimal alignments.
Alignments

- Optimal alignment for <Rice, Salmon, Salmon, Wasabi>:

\[
\lambda_{opt}^M(\sigma) = \begin{bmatrix}
\text{Rice} & >> & \text{Salmon} & >> & \text{Salmon} & >> & \text{Wasabi} & >>
\end{bmatrix}
\]

\[
\lambda_{opt}^M(\sigma) = \begin{bmatrix}
\text{Rice} & \text{Avocado} & \text{Salmon} & >> & \text{Wasabi} & \text{Eat}
\end{bmatrix}
\]

\[\delta(\lambda_{opt}^M(\sigma)) = 3\]

- Worst alignment:

\[
\lambda_{worst}^M(\sigma) = \begin{bmatrix}
\text{Rice} & \text{Salmon} & \text{Eat} & >> & >> & >> & >> & >>
\end{bmatrix}
\]

\[
\lambda_{worst}^M(\sigma) = \begin{bmatrix}
>> & >> & >> & \text{Rice} & \text{Avocado} & \text{Salmon} & \text{Nori sheet} & \text{Eat}
\end{bmatrix}
\]

\[\delta(\lambda_{worst}^M(\sigma)) = 8\]

- The fitness is defined as

\[
\text{fitness}(\sigma, M) = 1 - \frac{\delta(\lambda_{opt}^M(\sigma))}{\delta(\lambda_{worst}^M(\sigma))} = 0.625
\]
Alignments Discussion

• Alignments easier to understand: Instead of tokens in Petri-nets, we talk about skipped and inserted events.

• Higher accuracy, since Token Replay suffers from token flooding.

• Fitness values for Alignments tends to be to low, while Token Replay often yields higher values.

• More flexibility due to modifications of the costs $\delta$. E.g. activity "avocado" might be cheaper to drop than dropping the activity "rice".

• Not depending on Petri-nets only.

• However, very computational expensive.
Applications for Conformance Scores

• We only talked about conformance checking for fraud detection and workflow diagnostics.
• Fitness values determined by conformance checking provide us with a definition of distance between model and trace.
• The unstructured trace space, which is not a native vector space, becomes semi-metric.
  • The distance is not defined between traces, but uses models as reference points.
  • As the distance is not computed directly, but depends on a secondary structure, it is called geodetic.

• Using this distance, clustering and outlier detection become possible:
• We can also lift this approach to a log-to-log level, defining distances between two process logs for clustering and outlier detection (k-means, DBSCAN,…):
Temporal Conformance Checking

• Until now: Does the order of events conform to a given model? Often it is interesting if events are also executed at the "right" time.
• Even for conform traces, an activity can be executed too early or too late.
• In the following, the execution order was correct and according to model, there is no problem:

\[
\begin{align*}
&\text{cook rice} \rightarrow \text{prepare avocado} \rightarrow \text{salmon} \rightarrow \text{Combine and roll Nori sheet} \\
&6\text{h}13\text{m}12\text{s} \quad 0\text{h}2\text{m}43\text{s} \quad 0\text{h}4\text{m}7\text{s}
\end{align*}
\]

The last event failed due to dry and hard rice.

• Recent research on this at DBS:
1. Introduction

2. Basics

3. Supervised Methods

4. Unsupervised Methods

5. Process Mining
   5.1 Introduction
   5.2 Process Model/Transition Systems
   5.3 Process Discovery
   5.4 Conformance Checking
   5.5 Additional Mining Tasks
## Perspectives - Motivational Example

### Average daily outside temperature in °C

<table>
<thead>
<tr>
<th></th>
<th>Log 1</th>
<th>Log 2</th>
<th>RLE-based (Log 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>14.2</td>
<td>14.2</td>
<td>1*14.2</td>
</tr>
<tr>
<td>Day 2</td>
<td>14.4</td>
<td>14.4</td>
<td>4*14.4</td>
</tr>
<tr>
<td>Day 3</td>
<td>14.4</td>
<td>14.4</td>
<td>1*14.3</td>
</tr>
<tr>
<td>Day 4</td>
<td>14.4</td>
<td>-21.3</td>
<td>1*14.2</td>
</tr>
<tr>
<td>Day 5</td>
<td>14.4</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td>14.3</td>
<td>14.3</td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>14.2</td>
<td>14.2</td>
<td></td>
</tr>
</tbody>
</table>

Detecting anomalous behavior in temperature data by changing perspectives

**Log 1:**
E.g. Mean and standard deviation can be computed
⇒ still seems normal

**Log 2:**
Point anomaly is obvious

**Log 3** (Saves entries of Log 1 in a Run-Length Encoding manner):
Exposes entries of Log 1 as a possible collective anomaly
Motivation - Perspectives

• Analysis can be done by using different perspectives

=> Event logs provide much more information
E.g.: Timestamps, resources, transactions, costs etc.

• Thus far: Control-flow perspective

• Moreover:
  • Time perspective
  • Case perspective
  • Organizational perspective
Motivation - Perspectives

Time perspective
- Focus on timing and frequency of events
- Goals: Discover bottlenecks, monitor utilization of resources, remaining time prediction

Case perspective
- Focus on case properties
- Properties can be case attributes, event attributes, a path taken, performance information
- Goals: Mining decisions (e.g. a specific path) based on the characteristics of the case shows which data is relevant and should be included in the model

Organizational perspective
- Focus on information about resources
- Resources can be people, systems, roles, departments
- Goals: Classify actors in terms of roles, show social network
Temporal Visualization – Dotted Chart Analysis

- How to get a general overview: **Dotted Chart Analysis**

Classifier (here: *Case ID*) sorted by timestamp of first event

Dots correspond to events belonging to a specific class

Legend mapping event colors to event descriptors

Dotted chart for some receipt process

5. Process Mining
5.5 Additional Mining Tasks
5. Process Mining

5.5 Additional Mining Tasks

Temporal Visualization – Dotted Chart Analysis

- Shorted cases at the top
- Longer ones at the bottom

Created with ProM (Process Mining framework) with Dotted Chart Plugin

Time since case started sorted by duration of a case
Temporal Visualization – Dotted Chart Analysis

Time since week started.
Indicates that only few events were executed by night and at weekends.

→ Most events on weekdays between 9am and 4pm
Temporal Mining

Presence of timestamps enables

- **discovery of bottlenecks**
  - Limitation of capacity of a specific resource

- **monitoring of resource utilization**
  - *Which resources are occupied by which activity the most?*

- **prediction of remaining processing times of running cases**
  - Based on computations made on discovered cases so far

- etc.

Token replay can be extended to replay event logs with timestamps included (*time-based replay*).

This can help to extract aforementioned information.
Temporal Mining – Time-based replay

Replay of first part of our sushi process for two cases starting at 3pm i.e. 5pm

Timed replay for 2 cases showing durations at transitions and waiting times at places
Temporal Mining – Time-based replay

Replay of first part of our sushi process for two cases starting at 3pm i.e. 5pm

Partial sushiing process seems to have a bottleneck at \(\langle \text{cook rice, season rice} \rangle\)

Record collection of token visits → derive multi set of durations for each place
Temporal Mining – Time-based replay

Possibility to
• Fit distribution
• Compute statistics such as
  • mean,
  • standard deviation,
  • minimum,
  • maximum
  • etc.

⇒ Visualization of waiting times
⇒ Visualization of service times
⇒ Bottleneck detection and analysis

Timeline resembling a Gantt-Chart (excerpt of time-based replay)
Trace Clustering - Motivation

High *diversity*:
Single cases differ significantly from one another

→ possibly very complex models

Our sushiing process already can be very complex depending on the granularity of visualization.
Trace Clustering - Motivation

**Assumption:**

Process variants hidden within the event log

⇒ Cluster traces before discovering a model

⇒ Clustering approach also based on different perspectives

Example: Second part of our sushining process
Trace Clustering - Example

- How to determine a similarity value between our data points (here: cases)?
- Clustering on points in vector space is well-known

=> Embedding of cases into vector space necessary \(\rightarrow\) Profiles

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Roll and slice</th>
<th>Add wasabi</th>
<th>Add soy sauce</th>
<th>Prepare stir-fried rice</th>
<th>Eat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Add up the number of activity execution for each case

5. Process Mining
5.5 Additional Mining Tasks
Trace Clustering - Example

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Vector</th>
<th>Manhattan distance to other vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 1, 0, 0, 1)</td>
<td>2, 0, 2, 1</td>
</tr>
<tr>
<td>2</td>
<td>(1, 0, 1, 0, 1)</td>
<td>2, 2, 0, 3</td>
</tr>
<tr>
<td>3</td>
<td>(1, 1, 0, 0, 1)</td>
<td>0, 2, 2, 1</td>
</tr>
<tr>
<td>4</td>
<td>(1, 0, 1, 0, 1)</td>
<td>2, 0, 2, 3</td>
</tr>
<tr>
<td>5</td>
<td>(1, 1, 0, 1, 1)</td>
<td>1, 3, 2, 3</td>
</tr>
</tbody>
</table>

E.g. cluster with agglomerative approach

⇒ E.g. cluster with agglomerative approach
Trace Clustering – Methods

Aforementioned profile is called *Activity Profile (Activity Histogram)*
• Defines one item (feature) per type of activity
• An activity item is measured by counting all events of a trace which have that activities name
• Of course, **various other profiles possible** as well

In General:

*Profile*: Set of items with measurements
*Item*: Assigns numeric value to each trace

⇒ A Profile can be considered a function \( f \) which maps a trace \( t \) to a vector \((i_1, i_2, ..., i_n)\) with \( n \) items:

\[
f(t) \rightarrow (i_1, i_2, ..., i_n),
\]
Trace Clustering – Methods

More examples:

Transition profile:
**Items:** Direct following relations in a trace
**Measure:** How often an event A has been followed by an event B
**Goal:** Measure behavior of traces (capturing the context) cf. n-grams

Performance profile:
**Items:** Size of a trace regarding timestamps: case duration, (min, max, mean) time difference between events etc.
**Measure:** Depends on predefined items e.g. size is measured by number of events
**Goal:** Measure performance of a trace (→ also part of Temporal Mining)
Additional Mining Tasks - Roundup

- Processes can be analyzed by using different perspectives:
  - time
  - case
  - organizational

- Get an overview by applying **Dotted Chart Analysis**

- **Temporal Mining** useful to:
  - detect bottlenecks
  - monitor resource utilization
  - predict remaining processing time

- **Trace Clustering** helps to distinguish between process variants dependent on different perspectives (*profiles*)
Resources

• ProM Framework

