Refining Imprecise Labels

Xixi Lu
Dirk Fahland
Wil van der Aalst

Eindhoven University of Technology
Practical Problem and Motivation

Log with imprecise labels

$t_1 = s \text{ abc a e}$
$t_2 = s \text{ cba e}$
$t_3 = s \text{ cab e}$
Real-life Example: Hospital Data

3 tests and consult

1st surgery and consultation

3 tests and consult

1st surgery and consultation

[Diagram showing a process flow with activities like Consultation, Surgery, and Tests, along with decision points and parallel processes.]
2. Problem Setting

Our focus: Problem of label refinement

The optimal labels are unknown!
Aims and Complexity

• Refine labels to
  • (A1) help discover a model closer to the system
  • (A2) help discover more precise and structured models yet preserve fitness
  • (A3) provide users with more options and different views w.r.t. a log

• Complications

<table>
<thead>
<tr>
<th>Complications</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A1) The system is unknown and depends on the domain experts</td>
<td>Use heuristics</td>
</tr>
<tr>
<td>(A2) More precise log $\not\Rightarrow$ more precise model $\not\Rightarrow$ higher precision, depends on the algorithm and the precision metric used.</td>
<td>Have a backup plan $\Rightarrow$ Return the original log, so we are not worse</td>
</tr>
</tbody>
</table>
Complication w.r.t. A2

Our focus: Problem of label refinement

Refrining Label

Imprecise

\[ t_1 = \langle s \ abc \ a \ e \rangle \]
\[ t_2 = \langle s \ cba \ e \rangle \]
\[ t_3 = \langle s \ cab \ e \rangle \]

Precise

\[ \langle s \ a_2 b_2 c_2 a_3 e \rangle \]
\[ \langle s \ c_1 b_1 a_1 e \rangle \]
\[ \langle s \ c_1 a_1 b_1 e \rangle \]
\[ \langle s \ a_3 b_3 c_3 a_4 e \rangle \]
\[ \langle s \ c_2 b_2 a_2 e \rangle \]
\[ \langle s \ c_1 a_1 b_1 e \rangle \]
Research Problem

Imprecise log

S A B C T
S A B A B C T
S A B A B C T
S A A B T
S A C T
S A C T

Refining Label

“precise” log

S A_0 B_0 C_0 T
S A_0 B_0 A_0 B_0 C_0 T
S A_0 B_0 A_0 B_0 C_0 T
S A_1 A_2 B_1 T
S A_3 C_1 T
S A_3 C_1 T

Discover

Model such that ...
General Framework

0. Detect Imprecise labels

Imprecise log candidates
\{A, B, C\}

1. Refine Labels horizontally

Horizontal clusters of events

```
+-----+-----+-----+
| A   | B   | C   |
+-----+-----+-----+
| A   | A   | B   |
| A   | B   | A   |
| A   | A   | B   |
| A   | C   | T   |
| S   | A   | C   |
| S   | A   | C   |
+-----+-----+-----+
```

2. Refine Labels vertically

Vertical clusters of events

```
+-----+-----+-----+
| A   | B   | C   |
+-----+-----+-----+
| A   | B   | A   |
| A   | B   | A   |
| A   | A   | B   |
| A   | C   | T   |
| S   | A   | C   |
| S   | A   | C   |
+-----+-----+-----+
```

“precise” log

```
S A B C T
S A B A B C T
S A B A B C T
S A A B T
S A C T
S A C T
S A B C T
S A B A B C T
S A B A B C
S A B A B C T
S A B A B C T
```

Refining Label
Detect Imprecise Label Candidates

• Many ways...
  • 1) Oracle, known as domain expert
  • 2) Using inductive miner to parse log and find imprecise nodes
    e.g. (flower) loops, AND-splits/join
General Framework

0. Detect Imprecise labels

1. Refine Labels horizontally

2. Refine Labels Vertically

Mappings and costs
Mapping: Similar and Dissimilar Events

Trace 1: $S \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow E$

Trace 2: $S \rightarrow C \rightarrow B \rightarrow A \rightarrow E$

--- Similar = Mapped
--- Dissimilar = Not Mapped
Quantify Mappings

Using cost function to quantify dissimilarity

(1) Differences in k-neighbors
Quantify Mappings

Using cost function to quantify dissimilarity
(1) Differences in k-neighbors
Quantify Mappings

Using cost function to quantify dissimilarity
(1) Differences in k-neighbors
(2) Differences in structure

Distance = 4

Distance = 3
Quantify Mappings

Using cost function to quantify dissimilarity
(1) Differences in k-neighbors
(2) Differences in structure
(3) #Dissimilar events

\[
\begin{array}{cccccc}
S & A & B & C & A & E \\
S & C & B & A & E \\
\end{array}
\]
Quantify Mappings

- Using cost function to quantify dissimilarity
  - Optimal mapping captured most similar pairs

- Compute cost for the imprecise label candidates only

- Using existing algorithms
General Approach

0. Detect Imprecise labels

1. Refine Labels horizontally

2. Refine Labels vertically

Compute Mappings

Mappings and costs
1. Refine horizontally using cost (3D)

a) Normalize costs between 0 and 1
1. Refine horizontally using cost (3D)

- a) Normalize costs between 0 and 1
- b) Set cost threshold, e.g. 0.6
1. Refine horizontally using cost (2D)

- Trace 0
  - S → A → B → C → E
  - Costs: 0.5, 0.5, 0.5

- Trace 1
  - S → A → B → C → A → E
  - Costs: 0.5, 0.5, 0.75, 0.75, 0.75

- Trace 2
  - S → C → B → A → E
  - Costs: 0.5, 0.5, 0.5

- Trace 3
  - S → C → A → B → E
  - Costs: 0.5, 0.5, 0.5

Imprecise label candidates

- a) Normalize costs between 0 and 1
- b) Set cost threshold, e.g. 0.6
- c) Keep edges if cost ≤ cost threshold
1. Refine horizontally using cost (2D)

- **Trace 0**
  - S → A1 → B1 → C1 → E
  - Costs: 0.5 → 0.5 → 0.5

- **Trace 1**
  - S → A1 → B1 → C1 → A1 → E
  - Costs: 0.5 → 0.5 → 0.5

- **Trace 2**
  - S → C2 → B2 → A2 → E
  - Costs: 0.5 → 0.5 → 0.5

- **Trace 3**
  - S → C2 → A2 → B2 → E
  - Costs: 0.5 → 0.5 → 0.5

**Steps:**

a) Normalize costs between 0 and 1
b) Set cost threshold, e.g. 0.6
c) Keep edges if cost ≤ cost threshold
1. Refine horizontally using cost (3D)

a) Normalize costs between 0 and 1

b) Set cost threshold, e.g. 0.6

c) Keep edges if cost $\leq$ cost threshold
General Approach

0. Detect Imprecise labels

1. Refine Labels horizontally

2. Refine Labels vertically

Mappings and costs

S A B A B C T
S A B A B C T
S A A B T
S A C T
S A C T

Compute Mappings

{A, B, C}

A B C
A B A B C T
A B A B C
A A B
S A C T
S A C T

S A
S A

S A B A B C T
S A B A B C
S A A B
S A A B
S A C
S A C

S A 1
S A 2
S A 3
S A 3

S A 1 A 2 B 1 T
S A 0 B 0 A 0 B 0 C 0 T
S A 0 B 0 A 0 B 0 C 0 T
S A 1 A 2 B 1 T
S A 3 C 1 T
S A 3 C 1 T
S A 0 B 0 C 0 T
S A 0 B 0 A 0 B 0 C 0 T
S A 0 B 0 A 0 B 0 C 0 T
S A 3 C 1 T
S A 3 C 1 T

S C B A E

Mapping
2. Refine labels vertically using Frequency

Unfolding threshold = 60%
2. Refine labels vertically using Frequency

Unfolding threshold = 40%
General Approach

0. Detect Imprecise labels

1. Refine Labels horizontally

Mappings and costs

S A B C T
S A B A B C T
S A B A B C T
S A B T
S A

{A, B, C}

S A B
S A B A B C T
S A A B
S A C
S A C

Maps:

S → A → B → C → A → E
S → C → B → A → E

S A B C T
S A B A B C T
S A B A B C T
S A B T
S A

S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C

S A B
S A B A B C T
S A A B
S A C
S A C
Local Refining - Using Inductive Miner

0. Detect Imprecise labels

Refine labels

Replace subtree

Discover

{A, B, C}
Evaluation (A1)

N: Number of visible transitions
P: number of imprecise labels
Evaluation (A1)

N: Number of visible transitions
P: number of imprecise labels
Evaluation (A1)

N: Number of visible transitions
P: number of imprecise labels
Evaluation (A1)

- N: Number of visible transitions
- P: number of imprecise labels

System Process tree

Generator

L_trs

L_lab

Relabeling Globally (ReGlo)

Relabeling Locally – IM (ReLoc_IM)

IM

M_trs

M_lab

M_ReGlo

M_ReLoc

Compute Behavior Recall & Precision

Practical UB_IM

Behavior Recall & Precision

Behavior Recall & Precision

Practical LB_IM
Evaluation (A1) Behavior Precision and Recall

• For each event, compare behavior in the system and in the discovered model.
Evaluation (A2) Model Precision

Relabeling Globally (ReGlo)

Relabeling Locally – IM (ReLoc_IM)

Compute Model Precision

Practical UB_IM

Model Precision

Model Precision

Practical LB_IM
EXP1) N20(x20),P4, (Oracle Detection)

- Number of models improved

<table>
<thead>
<tr>
<th>Model</th>
<th>TRUE</th>
<th>FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>IM</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>ReLocM</td>
<td>7</td>
<td>13</td>
</tr>
</tbody>
</table>
EXP1) N20(x20), P4, (Oracle Detection)

(A1) behavior recall and precision absolute improvements (IM)
EXP1) N20(x20), P4, (Oracle Detection)

(A1) behavior recall and precision absolute improvements (ILP)
EXP1) N20(x20), P4, (Oracle Detection)

• Absolute Model Precision Improvements
EXP1) N20(x20), P4, (Oracle Detection)

• Absolute Model Precision Improvements
EXP1) N20(x20), P4, (Oracle Detection)

- % of maximal possible improvements achieved
EXP1) N5(x5), P2(x4), (IM Detection)
N5(x5), P2(x4), Examples (working)
N5(x5), P2(x4), Examples (not working)
N10(x5), P2(x4), (IM Detection)
N10(x5), P2(x4), working and not
EXP3) N10(x5),P4(x4), (Oracle Detection)
N10(x5), P4(x4), working
N10(x5), P4(x4), not working

M_gen

M_impr

M_loc_IM
N10(x5), P4(x4), not working
N10(x5), P4(x4), not working

M_gen

M_impr

M_loc_IM
N10(x5), P4(x4), working, if change threshold

M_gen

M_impr

M_loc_IM
N15(x5), P4(x4), working

M_gen

M_impr

M_ref_IM
N15(x5), P4(x4), not working
Three Main Causes For Fail to Refine

• Edge cost thresholds and unfolding threshold too low or too high
Edge cost = 10

Edge cost = 7

Edge cost = 6

Edge cost = 3;

With edge cost = 3, refold by set unfold threshold back to 20. Flower loop.
Three Main Causes For Fail to Refine

• Edge cost thresholds and unfolding threshold too low or too high

• Loops too long

• Too many concurrent events
Future Work, Conclusion and Questions?

- More experiments...