User Manual

Label Refinement by Behavioral Similarity

Document owners:
Bianka Bakullari
Christopher Beine
Nicole Ventsch
Juan Garza

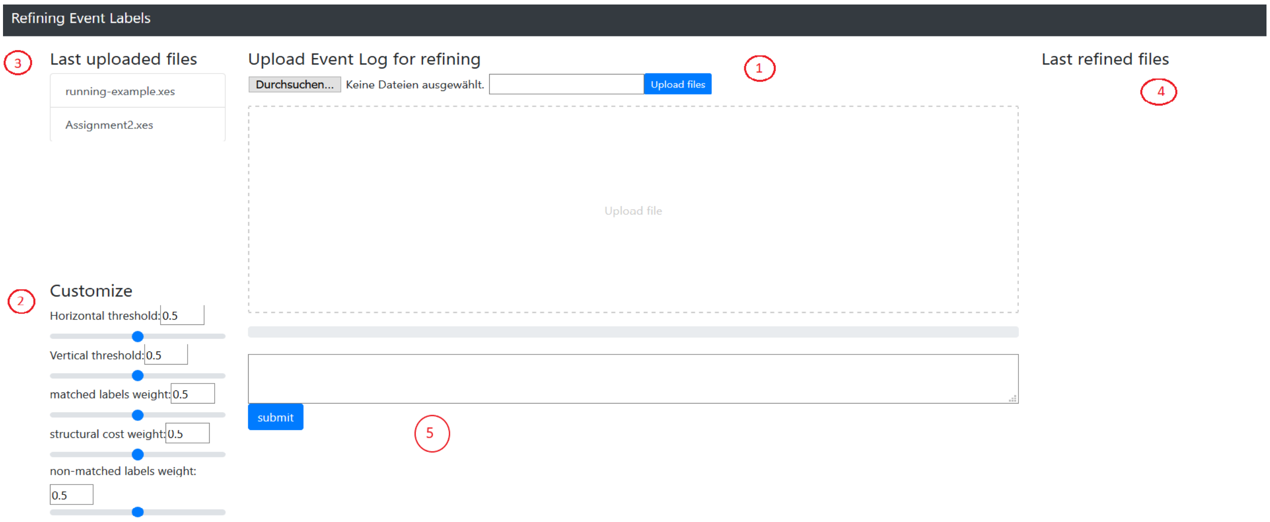
Last edited: July 4, 2019

1 Overview

Many processes involve carrying out an action multiple times. An example for this would be an online shop in which you first have to pay a registration fee before ordering an item and paying it. This process contains the event “payment” twice, but in different contexts, so that the payments are actually two different tasks. When analysing processes behind event logs, each event appearing in the log has a certain label as its *activity name*, so if both tasks above had the label “payment”, the process discovery algorithms would treat them as the same task, yielding models which contain loops. However, these loops do not match the actual process and make the model imprecise. This could be avoided if the tasks above were assigned different labels, e.g: ”payment1” and ”payment2”. Manipulating the event log by refinining its event labels could lead to more precise process models. The way the labels are refined is the issue this project addresses.
Using the web service we provide, imprecise logs are refined based on the structural contexts of the events. The refinement is executed by relabeling the original log and no filtering is applied to the log. Moreover, it is possible for the user to interactively change the thresholds used by the algorithm to influence the result. The approach we have implemented is described in [1].
In the following chapters, the usage of this web service will be explained.

2 Instructions

The start page of our web service consists of different parts, which will be explained in the following subsections. Namely, we have the parts ”Upload Event Log for refining” (1), ”Customize” (2), ”Last uploaded files” (3), ”Last refined files” (4) and ”Candidate Labels” (5), which can be viewed in Figure 1.



Horizontal threshold: 0.5

Vertical threshold: 0.5

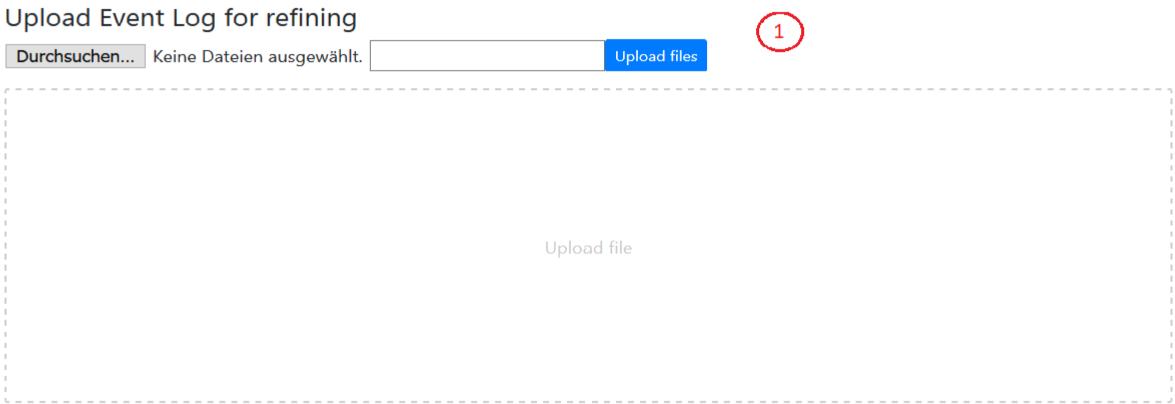
matched labels weight: 0.5

structural cost weight: 0.5

non-matched labels weight: 0.5

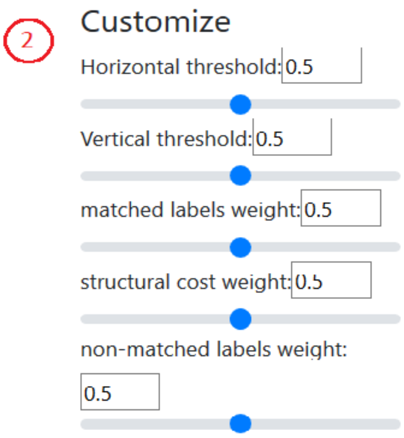
Figure 1: Overview of the start page with the different parts marked in red numbers

2.1 Upload Event Log for refining


Upload Event Log for refining

In this part of the start page, an event log can be uploaded. In order to do so, the user can either directly type in a path to a file and press the ”Upload files” button, or he can use the ”Search” button to search local files on his PC or drag a file into the big box with ”Upload file” written in it. Afterwards, he has to press the ”Upload files” button in order to upload the selected file.
In order to successfully upload a file, this file has to be in XES or CSV format, otherwise an error will occur.

2.2 Customize


Customize

Horizontal threshold: 0.5

Vertical threshold: 0.5

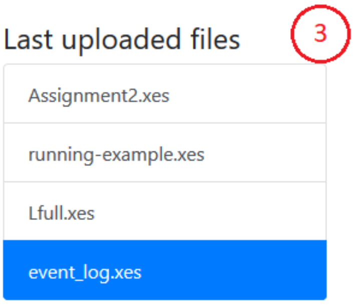
matched labels weight: 0.5

structural cost weight: 0.5

non-matched labels weight: 0.5

In this section, the user can adjust the thresholds in order to fit his needs. To do so, he can either adjust them using the provided slide control or by typing them manually. The thresholds the user can adjust are the horizontal threshold, the vertical threshold, the weight structure, the weight for matched pairs of activities and the weight for not matched pairs of activities. All of the thresholds and weights have to be in the range from 0 to 1. By default, all values are set to 0.5.

2.3 Last uploaded files



Last uploaded files

Assignment2.xes

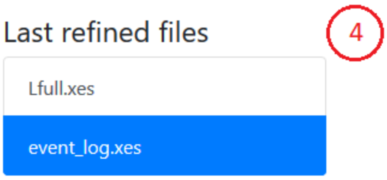
running-example.xes

Lfull.xes

event_log.xes

In this part of the start page, the user can see up to five of his last uploaded files. These files are not refined yet. By clicking on them, he can select them and apply the refinement algorithm on them.

2.4 Last refined files



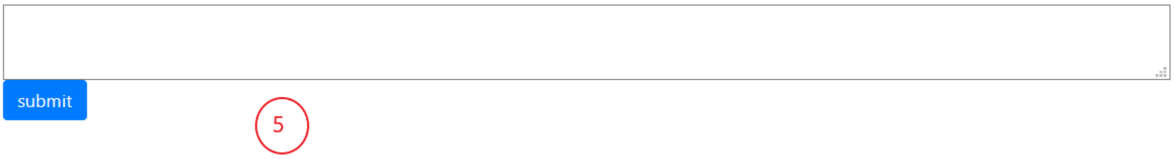
Last refined files

Lfull.xes

event_log.xes

In this part of the start page, the user can see up to five of his last refined files. These files are already refined. By clicking on them, the user can download them. The downloaded file has XES format and can be further used as input to process discovery algorithms.

2.5 Candidate Labels



In this section, the user provides the algorithm with the candidate labels that should be subject to possible refinements. To do so, he enters the labels into the white box. The format should be: $label_1$,$label_2$,...,$label_n$ and the order of the labels does not matter. After entering the candidate labels, the user can click on the "submit" button, so that the refinement algorithm will be run using the current thresholds, the chosen file and the provided candidate labels. The fresh labels have the form $labelx.y$ with x, y natural numbers. Here x identifies the refinement made in the horizontal refinement part of the algorithm, whereas y identifies the refinement made in the vertical refinement part. If no labels are given or the given labels do not appear in the log, the refined log will be identical with the original one.

3 Technical background

3.1 Frontend

The webapplication is optimized for Google Chrome (Version 65) and Mozilla Firefox (Version 60). The application has not been tested for Internet Explorer or other browser versions, so some parts may not work properly or the webinterface differs from the pictures. In order to access the API it is necessary to interpret the JSON object correctly.

3.2 Backend

The application backend can be executed on an Windows and Linux application server. This requires a Python version later than 3.6 and a Flask version later than 1.0. The index.py file in the application root dir indicates the entry point for the webapplication.

4 Glossary

In this section we give some brief description of the basic concepts used in the approach.

-task
This is a synonym for the word event. A task/event is unique in the log and it represents the recording of some action happening on a particular time. We usually identify an event by the activity name assigned to it. Many events (usually) have the same activity name describing ”what” happened.

-label
The labels stand for the activity names. An event label is equal to the activity name of that event.

-label refinement
By refining a label, we assign to it an other fresh label. By definition of a refining function, two distinct labels do not get assigned the same new label.

-relabel
A synonym for assigning a new name to a label.

-horizontal threshold
This threshold is important in the horizontal refinement part of the algorithm. For all pairs of traces in the log, we measure a value to capture the similarity (or dissimilarity) between them. Traces are considered similar if this value is under the horizontal threshold. The smaller the horizontal threshold, the more refinements take place.

-vertical threshold
This threshold is important in the vertical refinement part of the algorithm. In a set of traces which were considered similar in the horizontal refinement part of the algorithm, the same label may still appear many times. By adjusting the vertical thresholds one affects the strictness of the refinement computed on these labels. A lower threshold may induce more refinements.

-matched labels weight
If a pair of events appears in a mapping, this weight denotes the importance we give to the dissimilarity these two events have in the set of their predecessor and successor labels.

-structural cost weight
If a pair of events appears in a mapping, this weight denotes the importance we give to the dissimilarity these two events have in the positions they appear in the respective traces.

-non-matched labels weight
For an event not appearing in a mapping, the cost of the mapping regarding the event is increased by the product of this weight with the number of predecessors and successors of this event.

5 Further improvements

In this section we give some recommendations on how the algorithm can be adjusted or improved.

• Instead of feeding the set of all variants to the refinement steps, one could remove those variants that do not contain any candidate labels. This does not change the logic behind the approach. However, in the case that the mapping with the maximal cost is one between two filtered variants, the normalized weights of the graph are going to be affected. One can still obtain the same refined logs, but the corresponding thresholds yielding them would be slightly different.

• In our implementation, we compute the context of each event by looking at the set of all its predecessors and successors. One can change this to consider only the neighbors with maximal distance k for some arbitrary k.

Moreover, one could add this parameter to the other Custom parameters, so that a user that has a good domain knowledge regarding his event log can determine the scope k.

• For event logs containing a high number of trace variants or have long trace lengths which contain candidate labels, the algorithm has a long runtime. This is due to the fact that in our implementation we calculate the costs of all the mappings between all pairs of traces and then pick the best one. The calculation of the mappings can be further improved in a way that not all mappings are considered, but the difference in the results is negligible.

References

[1] Lu, Xixi, et al. "Handling duplicated tasks in process discovery by refining event labels." International Conference on Business Process Management. Springer, Cham, 2016.

[2] La Rosa M., Loos P., Pastor O. (eds) Business Process Management, BPM 2016, Lecture Notes in Computer Science, vol 9850. Springer, Cham, 2016.