Interactively Exploring Logs and Mining Models with Clustering, Filtering, and Relabeling

Xixi Lu, Dirk Fahland, Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
{x.lu,d.fahland,w.m.p.v.d.aalst}@tue.nl

Abstract. Real-life event logs often contain many data quality issues, which obstruct existing discovery algorithms from discovering meaningful process models and process analysts from conducting further process analysis. In this paper, we present an integrated tool that provides support for dealing with three of these data issues: logs comprising recordings of multiple heterogeneous variants of a process; traces containing multitude of deviating events in an infrequent context; event labels being imprecise. The tool is called *Log to Model Explorer* and helps users in interactively and iteratively exploring and preprocessing a log by clustering, filtering and event relabeling, enabling them to discover more meaningful process models.

Keywords: Process Mining, Log Exploration, Log Preprocessing, Trace Clustering, Log Filtering, Duplicated Tasks, Event Label Refinement, Process Discovery

1 Exploring Log for Complementing Process Discovery

Process Mining takes as input an event log that contains event data about past process executions. Real-life event logs may have many data quality issues. For example, they often contain various heterogeneous variants of the process. Every case going through the process may have some deviating events. Moreover, events may have imprecise and ambiguous labels. These data quality issues cause process discovery algorithms to fail in extracting meaningful process models. Although many methods and techniques have been proposed for handling some particular data quality issues, no integrated, explorative approach and support for preprocessing an event log has been proposed yet.

We discuss the use case using the the Road Traffic Fine log [1]. A user may start with applying a discovery algorithm on the log. However, due to aforementioned data quality issues, the user could obtained a rather complex model unsuitable for understanding the process. Fig. 1 shows a discovered model which contains a large loop in which each activity can be either executed or skipped, suggesting that in this part activities have been executed arbitrarily. However, there is no (set of) traces in the log showing this arbitrary behavior. Rather the log contains multiple different variants that are shown in Fig. 4. In order to identify these variants, the user would now have to pre-process the

Fig. 1: Model discovered with Inductive Miner from a log with data quality issues.

log (by choosing from various techniques), then discover a model and evaluate whether the model is good or further/different preprocessing is required. User would have to try dozens clustering plugins, filter the log in various ways and apply multitude discovery algorithms, not to mention the parameters he/she need to set when trying them. This makes exploring a log a rather laborious task, especially if the user does not know where the issue lies and the methods he needs may be distributed over different tools. Support for interactively exploring a log for discovering suitable models is missing.

This paper describes the plugin *Log to Model Explorer* in the *TraceMatching* package of the Process Mining framework ProM[2]. The plugin allows users explore different views on a log by integrating the support for three main preprocssing functionalities: clustering traces into variants, filtering infrequent behavior, and refining imprecise labels. The tool immediately shows the result of each preprocessing step by for example visualizing the model discovered. This allows user interactively and iteratively explore the log and discover models of interest. It integrates our previous work on clustering, filtering [2] and relabeling techniques [3] for control-flow behavior.

The aim of this demo paper is to demonstrate the basic concepts of the *Log to Model Explorer* and illustrate its general functionality and suitable use cases. We demonstrate our ideas on a random sample of 1000 traces of the Road Traffic Fine (RTF) log [1]. A screencast which demonstrates the features of the plugin and how it can be used in practice using the same log can be downloaded[3]. In Sect. 2, we first give an overview of the tool and explain the three main functionalities: clustering, filtering and relabeling. In the end, we discuss related tools and future work as we conclude.

2 The *Log to Model Explorer*

Our tool integrates three pre-processing operations on the log (clustering, filtering and relabeling) with immediate discovery of a model on (parts of) the pre-processed log. Fig. 2 shows an overview of possible paths comprising these four steps. Given an input log, the tool starts with clustering the traces based on their behavioral similarity (1). By selecting a cluster, the user is shown the corresponding sublog or the model discovered on this sublog (4). Optionally, the user may apply a filter on this sublog to remove infrequent events (2) and view the result of such filter (4). Next, for a cluster, the user may choose to refine event labels (3) while exploring different representations of models (4). After refining the labels, the user may decide to cluster the relabeled sublog again, and so on, iteratively. Fig. 3 shows the main GUI of the tool, highlighted with the enumerated sections that correspond to the aforementioned steps. Both logs and models at any stage can be exported for further analysis.

[2] Availabel in the ProM nightly builds under: `http://promtools.org`

[3] `https://svn.win.tue.nl/repos/prom/Documentation/TraceMatching/2016demo.mp4` or watch online at `https://vimeo.com/176721009`

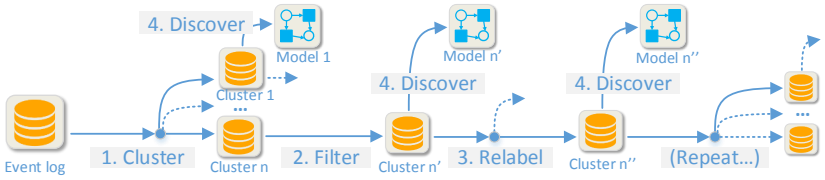

Fig. 2: Possible interactive exploration paths in the *Log to Model Explorer*.

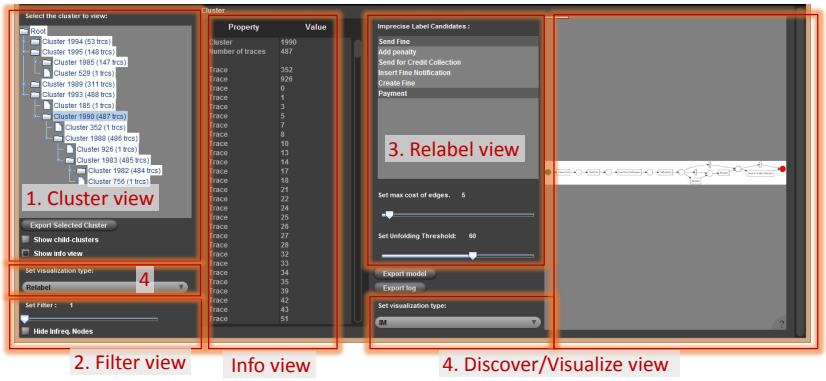

Fig. 3: The main GUI of the *Log to Model Explorer* plugin in the ProM Framework.

Trace Clustering. Trace clustering is an essential step of log-preprocessing to identify variants of the process in the log that are behaviorally similar [4]. A sublog of behaviorally similar traces is more likely to yield a more coherent and structured model. We use the clustering technique of [2] that measures behavior similarities of traces and builds a hierarchy of clusters, which we visualize as a tree in the cluster view (1) as shown in Fig. 3. The root node has a set of main clusters as its children. Each main cluster can be further expanded to view its sub-clusters. The user may select any (sub-)cluster and explore the cluster further. When a user selects a cluster, the plugin invokes the current visualizer, for example the Inductive Miner [5], which immediately shows the model descovered that represents this part of the log. It is also possible to view the selected cluster and its child-clusters together by checking the radio-button "*Show child-clusters*". For example, Fig. 4 shows that the user selected cluster 1982; the right-hand side of the screen shows at the top the model discovered for cluster 1982 and below the models for the two children of cluster 1982, i.e., the sub-variants. When inspecting the two sub-clusters, we see that the first one shows multiple executions "*Payment*" (in a loop) whereas the second sub-cluster shows no "*Payment*" has been executed; the model of the parent cluster 1982 combines both into a loop that can be skipped.

Context-Aware Event Filtering. Next, the user may decide to filter *infrequent behavior*, sometimes interchangeably called *deviations* or *noise*, from the traces of a cluster to reveal the main behavior of the cluster. This filtering technique is based on the deviation/noise detection described in [2] which does not requiring any normative process model. We consider an event e of activity A *infrequent* (uncommon, deviating) if e has a context of preceding and succeeding events that is different from most other occurrences

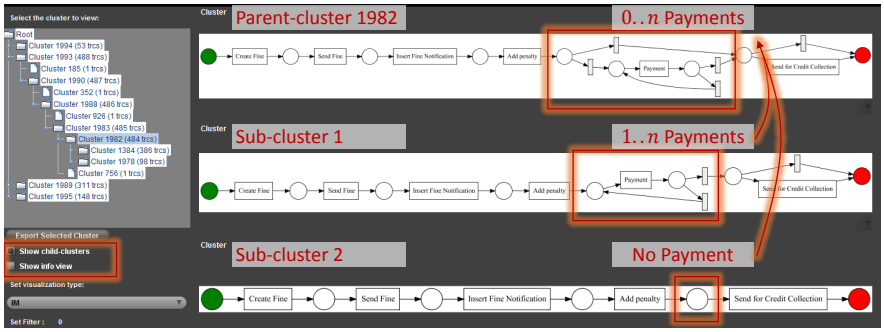

Fig. 4: Interactive, visual comparison of a cluster and its sub-variants.

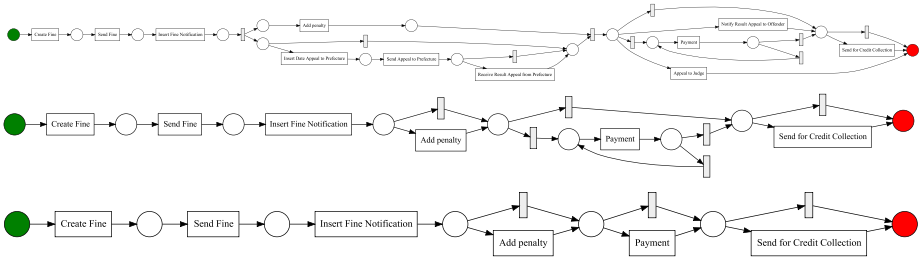

Fig. 5: The models discovered by filtering events based on the frequency of their context for 0%, 1% and 5% threshold.

of activity A. For example, assume a log with 99 traces $\langle abcd \rangle$ and 1 trace $\langle abdbde \rangle$; activity b has two contexts, between a and c, and between c and d; the latter one is infrequent. In a similar way, the single occurrence of e can be classified as infrequent. Our plugin provides a filter to remove all such infrequently classified events below a user-chosen threshold. The filtered log only shows the frequent behavior. Selecting a cluster (1993) in the RTF sample, we respectively obtain three models shown in Fig. 5 by filtering the cluster by 0%, 1% and 5%. Note that when setting the threshold to 5%, the loop around "*Payment*" is removed, revealing that less than 5% of the cases have a second "*Payment*".

Event Relabeling. In addition to clustering and filtering, we also provide support for refining event labels. Relabeling events allows user explore different representations of the same log, in particular when events of the same activity have clearly different contexts that are equally frequent. For example, assume the traces $\sigma_1 = \langle abcd \rangle$ and $\sigma_2 = \langle abcbde \rangle$ are equally frequent. Rather than filtering out the second b in σ_2, the user may want to view and analyze all behavior in the log. Giving the log as-is to a discovery algorithm would introduce a loop for repeated execution of b, but also several skip steps to allow d to occur after c (as in σ_1) and after b (as in σ_2), thus introducing many more behaviors. Alternatively, our plugin detects that the second b in σ_2 has a different context and allows to relabel it to b_2 allowing to discover a more precise model [3]. The user can influence the amount of relabeling by threshold parameters. Together with a discovery algorithm that guarantees discovering a fitting model, the user

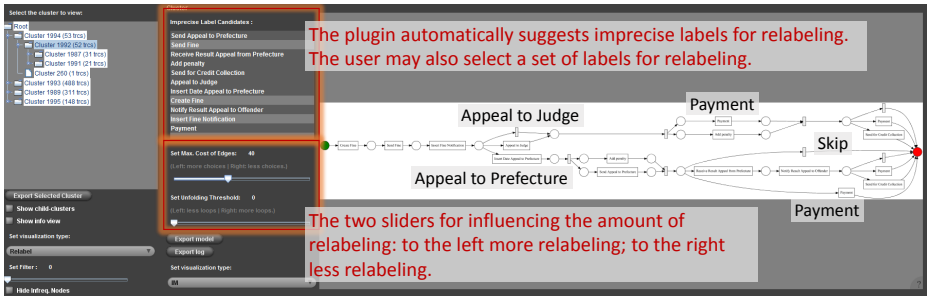

Fig. 6: Identifying two different variants within a process through event relabeling.

may now explore different models for the (sub)log that are all fitting but have different precision (generalization). Taking the RTF sample, the relabeling allowed us to find the duplicated tasks ("*Payment*", "*Add pentaly*" and "*Send for Credit Collection*") and to discover two alternative paths in the model as shown in Fig. 6: if offenders decide to "*Appeal to Judge*", then there is always a "*Payment*"; if offenders decide to "*Appeal to Prefecture*", then there are alternatives to skip "*Payment*". User may export discovered models, relabeled logs or original logs of cluster in ProM for further analysis.

Conclusion, Limitation and Future Work. In this paper, we presented the *Log to Model Explorer* as an integrated tool for log preprocessing and discovering more suitable models for a log. We showed that the tool supports three main functionalities: trace clustering, infrequent event filtering, and event label refinement. Currently, commercial tools such as Disco and Celonis[4] have extensive support for filtering a log for obtaining a variant but require the user to know which variant he/she want to have. Furthermore, these tools also suffer from the imprecise label problems and often discover spaghetti-like process maps. Available academic tools focus on solving one particular data quality issue; using them iteratively is tedious. A limitation of our tool is its performance in handling large logs; the running time scales polynomial in number of events. Currently, random sampling is used for improving the performance. As next step, we plan to generalize the tool into a framework for exploring an event log, allowing different clustering, filtering and log visualization techniques to be integrated.

References

1. de Leoni, M., Mannhardt, F.: Road Traffic Fine Management Process. Technical report, Eindhoven University of Technology (2015)
2. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Detecting deviating behaviors without models. In: BPM 2015, Workshops, Springer (2015) (to appear)
3. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Handling duplicated tasks in process discovery by refining event labels. In: BPM 2016, Springer (2016) (to appear)
4. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8) (2006) 1010–1027

[4] Disco :`https://fluxicon.com/disco/`; and Celonis : `http://www.celonis.de/`

5. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Application and Theory of Petri Nets and Concurrency. (2013) 311–329