Process Pruner: A Tool for Sequence-Based Event Log Preprocessing

David Baumgartner, Andreas Haghofer, Martin Limberger
Department of Data Science and Engineering
University of Applied Sciences Upper Austria
4232, Hagenberg, Austria
{firstname}.{lastname}@students.fh-hagenberg.at

Emmanuel Helm
Advanced Information Systems and Technology
University of Applied Sciences Upper Austria
4232, Hagenberg, Austria
emmanuel.helm@fh-hagenberg.at

***Abstract*—A major challenge in applying process mining on real event data is the presence of noisy or incomplete cases or unusual behaviors. Applying process mining on raw event data leads to wrong conclusions during the discovery of process models, concealing the typical behavior. In this paper, an alternative for filtering event data without the need for extensive preprocessing is presented. The method is based on generated footprint matrices of randomly pruned sub-logs and works in a semi-automated manner. By identifying the most similar matrices to validate the whole log, traces representing unusual behavior can be excluded or highlighted. The tool was implemented with Python 3, NumPy and Pandas and is publicly available on GitHub. We evaluated our tool using benchmark data-sets and compared it to human filtering and discovery results.**

***Index Terms*—Data mining; Process mining; Preprocessing**

I. Introduction

Process mining represents an essential and evolving part in the field of data analysis. It provides "fact-based insights and supports process improvements of business processes" [1]. With techniques provided by the field of process mining, it is possible to extract knowledge from event logs which are produced everywhere in todays information systems [2], [3]. In this context, a trace is a particular sequence of events beginning with a start-event and ending with an end-event. Real-life event logs usually include some noise. This is the result of rare and infrequent behavior in processes or can be produced by miss recordings or through errors. A cleaning task is therefore mandatory to be able to discover and extract useful knowledge. This step is part of the preprocessing in the data analysis context. The goal can vary from highlighting or temporarily excluding unwanted cases [4]–[6].

This proposed tool and its algorithm represent an approach for an automated cleaning task, which can find the most frequent control flows in an event log. The whole technique is based on the concept of footprint matrices, which were introduced with the alpha algorithm. [1]. This algorithm extracts information for each trace and creates so-called footprint matrices for further analysis. They represent the event flow for one or more cases. These matrices can be used to find patterns and similarities in the control flow of the event log.

This method can clean event logs from unexpected behavior (i.e., rare sequences of events), it additionally allows inverting the result and highlight cases that represent rare or infrequent behavior.

II. Material

A combination of different software tools and libraries were used to create the Process Pruner tool. The actual program was created with Python 3.6 using the NumPy[1] and the Pandas[2] library. To visualize the results, ProM version 6.8 and Disco were used. Especially the included *Inductive visual Miner* [7] in ProM provided the necessary optical information to compare the generated results of the algorithm with the raw results.

III. Methodology

Before the actual pruning algorithm starts grouping the traces as seen in Fig. 1, an additional preprocessing can be applied. This stop targets the end events of every trace.

If the user provides a list of valid end events, then this information is used as a preprocessing filter before the actual algorithm starts. Each matrix is generated out of K amount of traces (default 50) which are generated by grouping all traces of the log by their identifier. After shuffling the resulting identifiers, K of them are chosen to constitute a sub-log. This process is repeated N times till N amount of footprint matrices are generated. The generation follows the rules described within the alpha algorithm [1]. Each of the matrices represents one way of how to interpret the core process of the event log. To figure out which of them are representative for the real world process, the similarity of every footprint matrix to all others is calculated. Based on the hypothesis, that the most similar ones are the best match to represent the core process of the event log. The similarity between the matrices is calculated by merely counting the identical entries and calculating the relative amount of them compared to the number of matrix-entries, this is shown in equation 1.

$$similarity = \frac{count(similarEntries)}{count(entries)} \tag{1}$$

[1]http://www.numpy.org/
[2]https://pandas.pydata.org/

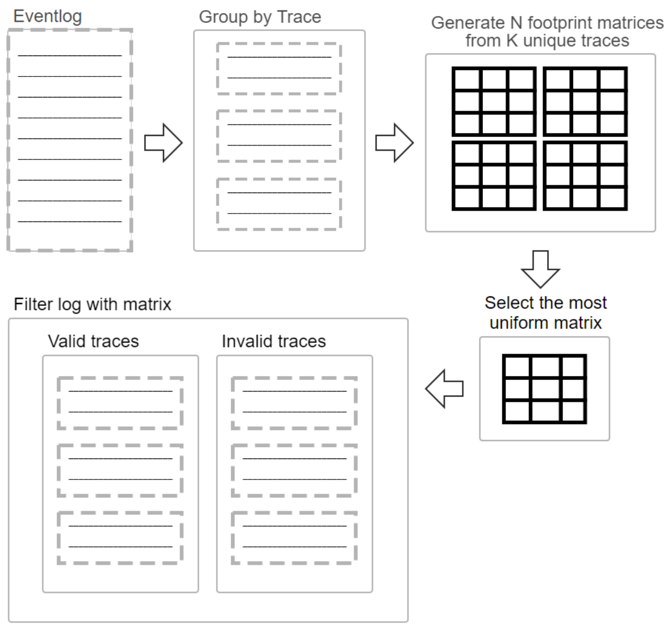
Eventlog
Group by Trace
Generate N footprint matrices from K unique traces
Select the most uniform matrix
Filter log with matrix
Valid traces
Invalid traces

Fig. 1. All cases of the log are group depending on their case id to get the whole trace. For each of the N generated footprint matrices, K traces are randomly selected (i.e. each trace is only used for the generation of exactly one footprint matrix). After calculating the similarity of each matrix to all the others, the most uniform one is used to validate the event log. The final log only contains the traces which could be replayed by the selected matrix.

This method works under the assumption that footprint matrices contain pairs of directly following events. Based on pairs, a full footprint matrix represent a sequence of pairs which express a possibility of the underlying event log [8]. Finally, this method requires more frequent behavior with correct sequences than strong varying ones. Therefore it is possible to extract the core process by selecting the most similar footprint matrices because these are the ones with less noise or invalid traces. The actual tagging is done by replaying each trace against one or more chosen footprint matrices. If this is possible with all of the selected footprint matrices, it is marked as a valid trace. Otherwise, it is dropped out of the final log. This leads to a generalized process model of the processed event log.

IV. Result

This tool with its algorithm can be used in a pipeline, where first filtering should happen to gain a better overview. This tool contains an interactive command-line interface, which allows to define the source CSV-file, the output-files and setting the trace and event identifier. Additionally, it allows filtering before the primary analysis by an end event criteria, which can be skipped. The output contains two files – one containing the matching traces and one the non-matching traces. The tool is also publicly available on GitHub[3] with a demo video[4].

[3]https://github.com/2er0/ProcessPruner

[4]https://2er0.github.io/ProcessPruner/ICPM-2019-Process-Pruner.webm

V. Evaluation

The proposed method is tested with two publicly available benchmark datasets. The tool needs a CSV-file for input. *eXtensible Event Stream*-files (XES) must be converted to a CSV-file, e.g. via ProM or Disco. For the evaluation, the default value 50 was used for the variable K.

a) Road Traffic Fine Management Process [9]: The first dataset is a real-life event log from a management system for road-traffic fines of an Italian local police force. This dataset has its specialties with a lot of incomplete traces and contains 150.370 traces comprising 561.470 events. The amount of events per trace varies from 2 to 20. The log without any preprocessing contains 231 different kinds of process variants as seen in Tab. I.

By applying a variant filter only keeping the traces that appeared more than once, 131 different variants of traces remained. The original event log contains lots of irregularities, i.e. there are many different process variants [10]. After manually filtering, only 8 variants are left, which represent the most essential processes. This is done by defining which end events are allowed and by removing incomplete traces, but requires insights into the data and domain knowledge. The Process Pruner automated apporach reduced the event log to 29 variations, which is almost like manual filtering.

The results between the different preprocessing can be seen in the Tab. I. The Process Pruner almost achieved the filter quality like manual filtering without any insights into the dataset.

TABLE I
THE RESULT AFTER APPLYING A VARIANT FILTER, MANUAL FILTERING AND THE PROCESS PRUNER

	Variants	Traces	Events
Original	231	150370	561470
Variant Filter 56%	131	150270	560551
Process Pruner	29	146147	534594
Manual Filter	8	121833	470119

b) Artificial Event Log - Hospital example [11]: This dataset provides event logs of medical processes with different amount of noise within the log. This data was originally used for a publication [12] were they showed a heuristic approach to reduce the noise within a log and not removing real behavior. Without any preprocessing, the final model as footprint matrix looks like Tab. IV.

As seen in the Tab. IV, there are more than one ways of how to start the process. This should not be possible, because the log is generated. As described in [12], there should always be ”Triage” at the beginning of each trace like in the real world. Therefore it can be noticed that the log contains traces which are incomplete in the way that the first few events are missing [13]. Without any parameters set, the presented algorithm is capable of solving this problem as seen in Tab. V.

This example clearly shows the potential of this automated preprocessing algorithm. Without any parameters set it was possible to filter incomplete traces and reduce the log in a way that the core process becomes visible. The Tab. II shows metric information for this event log. The original metric is therefore set against filtering by 80 percent variant with Disco and filtering with the Process Pruner. The table shows that the variant already reduced the most varying traces. The Process Pruner was able to remove more and was able to keep the primary process which is shown in the Tab. V.

TABLE II
THE RESULT AFTER APPLYING A VARIANT FILTER AND THE PROCESS PRUNER ON THE DATASET WITH 0 PERCENT NOISE

	Variants	Traces	Events
Original	346	100000	894708
Variant Filter 80%	279	99891	893566
Process Pruner	219	97506	880491

To prove that this is also possible with a noisy dataset, the used dataset, therefore, provides the base event log with 30 percent noise. The Tab. III contains the metrics of the original dataset and the variant filtering and the filtering with the Process Pruner. Our tool was able to find the primary process and removed about 50 percent of variants.

TABLE III
THE RESULT AFTER APPLYING A VARIANT FILTER AND THE PROCESS PRUNER ON THE DATASET WITH 30 PERCENT NOISE

	Variants	Traces	Events
Original	2946	100000	924239
Variant Filter 68%	1860	98914	912417
Process Pruner	1425	94252	874802

As seen in Tab. VI it is also possible to reduce noise within a log and get an approximation of the core process. Even if there are some differences compared to the previous example, it was possible to reconstruct parts of the core process. This includes the information that every trace in this context has to start with the ”Triage” event and the ”Organize Ambulance” event could only be placed after all the other ones. It is also clear that ”X-Ray” and ”Check” are compulsory events.

This example can be seen as a demo of how the presented algorithm can be used. If the user has to manually filter the invalid traces out of a dataset with 30 percent wrong events, it is for sure not as easy as to use this algorithm which could be run more than one time to get more and more stable results. Even with noisy data, it is possible to get an insight into how the real world process could be.

VI. DISCUSSION

Compared to a manually done preprocessing, the presented workflow clearly shows its strength in the field of filtering incomplete traces and noisy event logs. Especially for event logs with a high amount of events, it is not trivial to get an insight which traces are valid and which ones are not. Therefore it is time and cost saving to use preprocessing algorithms like the one presented in this paper to get stable results which deliver a good approximation of the real world processes.

Even data logs with a high amount of noise could be used to discover the core processes hidden in the log. This also delivers the possibility to isolate the noise producing parts of the real world by merely applying the algorithm in the way that not the most similar footprint matrices were used, instead of the more unique ones, which are representative for the noise and unusual behavior within the log. The combined information of the core process and the error producing and unique process instances deliver very detailed insight into the real world processes which could otherwise only be possible by investing much time by manually extracting the information out of the event log.

VII. CONCLUSION AND OUTLOOK

This tool was designed and implemented by students in the Process Mining class of the Data Science and Engineering Master’s program at the University of Applied Sciences Upper Austria in Hagenberg. It will be used in future lectures to compare the students’ manual data clearing efforts with an automated approach and to analyze the benefits and weaknesses of automated log filtering. With the next release, we plan to support XES-files as sources for this tool and plan to integrate it into *PM4PY*.

As the importance of process mining as well as the amount of collected log data rises, new preprocessing approaches are needed. We showed that our automatic preprocessing tool could help to improve the quality of logs by filtering incomplete or rare traces. Furthermore, we showed that the Process Pruner could be used to detect rare behavior, which might be interesting in scenarios where the rare traces have a significant impact on the process. More research has to be done in this area to make the preprocessing more accurate in terms of finding the most exciting cases.

REFERENCES

[1] W. M. Van der Aalst, “Process discovery: An introduction,” in *Process Mining*. Springer, 2011, pp. 125–156.

[2] W. Van der Aalst, “Data science in action,” in *Process Mining*. Springer, 2016, pp. 3–23.

[3] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs *et al.*, “Process mining manifesto,” in *International Conference on Business Process Management*. Springer, 2011, pp. 169–194.

[4] R. S. Mans, M. Schonenberg, M. Song, W. M. van der Aalst, and P. J. Bakker, “Application of process mining in healthcare–a case study in a dutch hospital,” in *International joint conference on biomedical engineering systems and technologies*. Springer, 2008, pp. 425–438.

[5] A. Bernstein, F. Provost, and S. Hill, “Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification,” *IEEE Transactions on knowledge and data engineering*, vol. 17, no. 4, pp. 503–518, 2005.

[6] D. Tanasa and B. Trousse, “Advanced data preprocessing for intersites web usage mining,” *IEEE Intelligent Systems*, vol. 19, no. 2, pp. 59–65, 2004.

[7] S. J. Leemans, D. Fahland, and W. M. van der Aalst, “Process and deviation exploration with inductive visual miner.” *BPM (Demos)*, vol. 1295, no. 46, p. 8, 2014.

[8] W. van der Aalst, “Process mining: Overview and opportunities,” *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 2, pp. 7:1–7:17, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2229156.2229157

[9] De Leoni, M. (Massimiliano) and Mannhardt, F. (Felix), “Road traffic fine management process,” 2015. [Online]. Available: https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

[10] F. Mannhardt, M. De Leoni, H. A. Reijers, and W. M. Van Der Aalst, “Balanced multi-perspective checking of process conformance,” *Computing*, vol. 98, no. 4, pp. 407–437, 2016.

TABLE IV
Footprint matrix representing the process in the event log without any preprocessing. This footprint matrix shows undesired behavior, e.g., there exist multiple ways to start a trace. For this dataset bypassing the ”Triage” event is a big exception and should not occur [12].

	Triage	Register	Check	X-Ray	Visit	Final Visit	Prepare	Organize Ambulance
Triage	#	>	∥	#	>	∥	<	#
Register	<	#	∥	>	∥	∥	∥	#
Check	∥	∥	∥	∥	∥	∥	∥	∥
X-Ray	#	<	∥	#	∥	∥	<	>
Visit	<	∥	∥	∥	#	>	∥	<
Final Visit	∥	∥	∥	∥	<	#	>	#
Prepare	>	∥	∥	>	∥	<	#	>
Organize Ambulance	#	#	∥	<	>	#	<	#

TABLE V
Footprint matrix of an event log preprocessed by the Process Pruner tool. Within this footprint matrix, every trace starts with the ”Triage” event followed by the ”Register” event before reaching other events. It is apparent that the algorithm removed incomplete and rare traces where this is not the case.

	Triage	Register	Check	X-Ray	Visit	Final Visit	Prepare	Organize Ambulance
Triage	#	>	#	#	#	#	#	#
Register	<	#	>	>	>	#	#	#
Check	#	<	∥	∥	∥	∥	∥	#
X-Ray	#	<	∥	#	∥	>	#	#
Visit	#	<	∥	∥	#	>	#	#
Final Visit	#	#	∥	<	<	#	>	#
Prepare	#	#	∥	#	#	<	#	>
Organize Ambulance	#	#	#	#	#	#	<	#

TABLE VI
Footprint matrix of preprocessed data log containing 30 percent noise. The Process Pruner is again able to find the correct start event. This matrix shows that the variability is still higher than in the matrix in table V but was able to reduce the noise and preserve the core process.

	Triage	Register	Check	X-Ray	Visit	Final Visit	Prepare	Organize Ambulance
Triage	#	>	#	#	#	#	#	#
Register	<	#	>	>	>	#	#	#
Check	#	<	∥	∥	∥	∥	∥	>
X-Ray	#	<	∥	#	∥	>	<	#
Visit	#	<	∥	∥	#	>	<	#
Final Visit	#	#	∥	<	<	#	>	#
Prepare	#	#	∥	>	>	<	∥	>
Organize Ambulance	#	#	<	#	#	#	<	#

[11] Mannhardt, F. (Felix), “Data-driven process discovery - artificial event log,” 2016. [Online]. Available: https://data.4tu.nl/repository/uuid:32cad43f-8bb9-46af-8333-48aae2bea037

[12] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, “Data-Driven Process Discovery - Revealing Conditional Infrequent Behavior from Event Logs,” in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, E. Dubois and K. Pohl, Eds. Springer International Publishing, 2017, pp. 545–560.

[13] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. van der Aalst, “Data-driven process discovery-revealing conditional infrequent behavior from event logs,” in *International Conference on Advanced Information Systems Engineering*. Springer, 2017, pp. 545–560.