Log Shifting for Enterprise Collaboration Systems: A Supervised Event Abstraction Approach (Extended Abstract)

Jonas Blatt*, Patrick Delfmann*, Florian Schwade* and Petra Schubert*
*Institute for IS Research, University of Koblenz-Landau
Universitätsstrasse 1, 56070 Koblenz, Germany
Email: {jonasblatt,delfmann,fschwade,petra.schubert}@uni-koblenz.de

***Abstract*—Event abstraction is a pre–processing method in Process Mining to create an interpretable log and, in turn, interpretable process models. Here, one or more events are aggregated to one event at a higher level of abstraction. The ProM plugin presented in this paper provides such an approach by training a log shifting model based on investigating a known high–level log and the related low–level log. This log shifting model can then be used to convert any low–level log of the same application. First evaluation results show that this approach is applicable. The log shifter presented in this paper is particularly tailored for Enterprise Collaboration Systems (ECS) that come with special low–level log characteristics.**

I. Introduction

Overly complex models, also referred to as spaghetti–models are a common problem in process discovery. They are hard to read and often have no further value for the business [1]. In such cases, pre–processing of the event log is necessary before a discovery algorithm is applied for mining a process model from the log. One suitable and promising approach is event abstraction, which aims to aggregate the events from a lower abstraction level (e. g., system logs or database operations) to events at a higher abstraction level [2]. By applying event abstraction, process discovery may result in a process model with fewer activities and process flow alternatives. Thus, the resulting process model is more comprehensible and better interpretable for humans.

In the last decade, several event abstraction approaches were developed [2]. For describing event abstraction, authors use terms such as Activity Mining [3], Event Log Abstraction [4], Activity Clustering [5], Log Lifting [6], and Sequence Clustering [1]. Some of these approaches address specific domains (e.g., Healthcare [7], Internet of Things [8], Customer Journey [9]), or address general business processes. Some of them consider intersections of low–level events, others do not. The approaches further differ in the underlying data mining techniques applied (e.g., supervised vs. unsupervised), the concrete family of algorithms applied, or the amount of human interaction required [2]. In general, existing approaches mainly use clustering algorithms (e.g. [4]) and/or machine learning approaches (e.g. [6]) to achieve the generation of the high–level log.

Social Process Mining (SPM) aims to mine rather unstructured collaboration processes from Social Software. As ECS support collaboration and communication in companies, they have emerged as the core components of the digital workplace in companies [10] and thus are valuable sources for mining and understanding collaboration processes. Due to the characteristics of ECS, event abstraction comes with special requirements. First, many ECS operate on multiple databases that react differently to user actions. One user action may result in multiple events in the low–level log generated by the system. Second, as the databases are independent from each other, the distance of the time stamps of two low–level events are not always the same and moreover, the low–level events of two corresponding high–level events might overlap temporally. Third, due to the same reason, the order of the low–level events might even vary slightly. Fourth, different sets of low–level events, which belong to the same high–level activity, may not contain the same low–level activities depending on the usage behaviour of the user. Fifth, there may exist low–level events, where no high–level event was triggered. Finally, logs generated by most ECS are not interpretable for data scientists. Hence, for generating interpretable logs, it is necessary to observe an ECS a sufficiently long time, record the user actions (e.g., by click path analysis) and thereby obtain a high–level log that can then be used to train a log shifter with the corresponding low–level log. Thus, we develop the log shifting tool presented in this paper which satisfies the outlined requirements for ECS logs.

The remainder of this paper is structured as follows: Section II contains an overview and description of the presented tool and section III presents the evaluation of the log shifting tool and an outlook on future work on event abstraction.

II. Tool Description

In a nutshell, we train a shifting model by examining known high–level traces and the corresponding low–level traces. The trained shifting model can then be used to shift an arbitrary low–level trace of the observed system to a previously unknown high–level trace. This approach is illustrated in Figure 1.

We use our approach in an ECS with more than 4,000 users in which we can observe and track user activity in the system using a tailored observer that records a high–level log representing the real business activities in the process. The

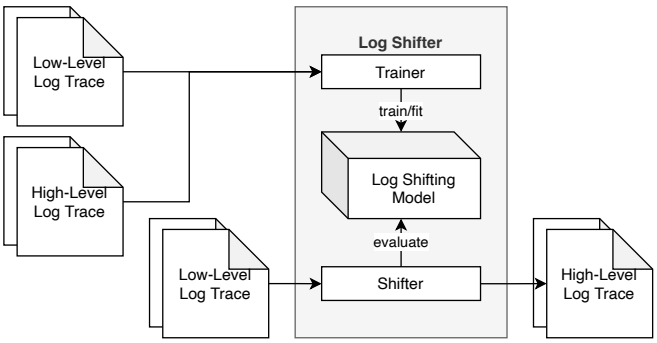

Fig. 1: Log Shifting Approach

observation is scheduled for 3 months for generating sufficient amount of traces. The observed high–level log can be used in combination with the low–level log generated by the system itself to create and train a new log shifting model. Then, this model can be used to convert arbitrary low–level logs from the system into high–level logs. We are able to *a)* shift the former low–level log (before we started the observation) or *b)* the low–level log from another instance of the same system.

In short, the algorithm works as follows: The shifting model contains the information about frequencies of the low–level events, which were mapped to the high–level events in the training phase (by temporal proximity & user equality). In the shifting phase, we use sliding windows, n–grams and the Damerau–Levenshtein distance [11] to calculate an error score for each possible combination of mapped windows of low–level events inside a trace. The windows of low–level events with the lowest error score are then mapped to new high–level events. This procedure repeats until there are no low–level events left. The new timestamp of the high–level event is calculated by the combination of the mapped low–level event timestamps (min, max, or mean).

The algorithm is implemented as a ProM[1] plugin, and can also be used as a command line program (see quick start instructions[2]). XES is used for the log format for the low–level log as well as for the high–level log. The generated log shifting model is a serializable Java class, which can be in– and exported to and from ProM. Thus, the workflow in the ProM plugin is as follows: Firstly, the low–level log, the high–level log and the training parameter are used to create the shifting model. Then, a low–level log and the previous generated shifting model can be used to create a new high–level log. A short introduction to the plugin can be found on YouTube[3].

III. Evaluation and Outlook

First evaluations with the BPIC20 logs[4] and synthetic created low–level logs with different configurations, corresponding to the typical characteristics of ECS low–level logs as introduced in section I, show, that this algorithm performs well and in a reasonable calculation time. For each activity in the original logs, we created one or more low–level activities (differs for each configuration). Then, the events in the logs were replaced by these low–level events. Thereby, we mix at random order these synthetic low–level events and adjust the mean temporal distance between the events. For each low–level log configuration, we trained multiple log shifting models (different training parameters) and calculated the evaluation accuracy, the train duration and the shift duration. In summary, this algorithm performs well (>=95% accuracy) and in a reasonable calculation time, if the training parameters are configured correctly regarding the log properties (can be automatically detected by hyperparameter optimization).

As outlined above, we are currently observing a high–level log from UniConnect, an ECS based on HCL Connections, which is the market–leading integrated ECS widely used in practice. In the next months, we will use the log shifting tool in the context of SPM to train a model to generate an interpretable log that is suitable for process discovery. The resulting model can be used for all instances of HCL Connections. In this future work, we consider more optimizations regarding the log shifting approach. For instance, we plan to implement hyper–parameter optimization routines for determining the best parameter configuration. We will also evaluate the applicability of the log shifting model to other ECS.

[1]https://www.promtools.org/

[2]https://uni-ko-ld.de/log-shifter-quick-start

[3]https://youtu.be/1Uo4yEK1UgI

[4]https://icpmconference.org/2020/bpi-challenge/

Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (Grant DE 1983/12-1)

References

[1] G. Veiga and D. Ferreira, "Understanding Spaghetti Models with Sequence Clustering for ProM," in *Lecture Notes in Business Inf. Processing*, vol. 43, 2009, pp. 92–103.

[2] S. J. van Zelst, F. Mannhardt, M. de Leoni, and A. Koschmider, "Event abstraction in process mining: literature review and taxonomy," *Granular Computing*, vol. 6, no. 3, pp. 719–736, 2021.

[3] C. W. Günther, A. Rozinat, and W. M. van der Aalst, "Activity Mining by Global Trace Segmentation," in *Computer Graphics Forum - CGF*, vol. 43, 2009, pp. 128–139.

[4] M. De Leoni and S. Dündar, "Event-log abstraction using batch session identification and clustering," *Proceedings of the ACM Symposium on Applied Computing*, pp. 36–44, 2020.

[5] J.-R. Rehse and P. Fettke, "Clustering Business Process Activities for Identifying Reference Model Components," in *BPM Workshops*, F. Daniel, Q. Z. Sheng, and H. Motahari, Eds. Cham: Springer International Publishing, 2019, pp. 5–17.

[6] G. Tello, G. Gianini, R. Mizouni, and E. Damiani, *Machine Learning-Based Framework for Log-Lifting in Business Process Mining Applications*. Springer International Publishing, 2019, vol. 11675 LNCS.

[7] A. Alharbi, A. Bulpitt, and O. Johnson, "Improving Pattern Detection in Healthcare Process Mining Using an Interval-Based Event Selection Method," in *Lecture Notes in Business Information Processing*, 2017, pp. 88–105.

[8] N. Tax, N. Sidorova, R. Haakma, and W. M. Van Der Aalst, "Mining Process Model Descriptions of Daily Life through Event Abstraction," *Intelligent Systems and Applications. IntelliSys 2016. Studies in Computational Intelligence*, vol. 751, 2018.

[9] G. Bernard and P. Andritsos, "CJM-ab: Abstracting customer journey maps using process mining," *Lecture Notes in Business Information Processing*, vol. 317, pp. 49–56, 2018.

[10] S. P. Williams and P. Schubert, "Designs for the Digital Workplace," in *CENTERIS - Conference on ENTERprise Information Systems*, vol. 138. Elsevier B.V., 2018, pp. 478–485.

[11] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. ACM*, vol. 7, no. 3, p. 171–176, Mar. 1964.