Statistical Sampling in Process Mining Discovery

Alessandro Berti

SIAV
Italy, 35030 Rubano (Padova)
Email: alessandro.berti89@gmail.com

***Abstract*—In this paper, we propose some ideas related to the application of Statistical Sampling techiques to simplify the application of a Process Discovery algorithm to big amounts of data. Much of the information about the business process could be indeed discovered by analyzing only a small amount of events, making useless the application of the algorithm to the entire data set.**

Keywords–Process Mining; Statistical Sampling; Big Data.

I. INTRODUCTION

Process Mining [1][2][3] is related to the discovery of information about business processes, and there are several techniques proposed for Business Process Discovery [4], Business Process Conformance [5], Business Process Prediction [6]. A recent field of research is about the application of Process Mining to big amounts of data [7][8][9]. We can cite some approaches to process discovery using GPU computing [10], Hadoop MapReduce [11], and an approach to streaming process data using Amazon Kinesis [12]. An open question is how much of the collected data is necessary for a Process Discovery algorithm in order to discover the process schema. In this paper, we propose some ideas about the use of statistical sampling to event logs that means applying a Process Discovery algorithm only on some process instances, e.g., a small amount of the collected data. The paper is organized as follows: Section 2 introduces a process discovery algorithm, Section 3 shows a method to apply statistical sampling to the process discovery algorithm introduced in Section 2, in Section 4 there are some measures in order to evaluate the soundness of the approach proposed in Section 3.

II. BACKGROUND

A widely used Process Discovery technique is Heuristics Miner [13][14]. The algorithm, given as input the event log, works calculating a dependency measure between activities:

$$dep(A,B) = |A => B| = \frac{|A > B| - |B > A|}{|A > B| + |B > A| + 1}$$

Where $|A > B|$ is the count of occurrences in the log where an event with activity A is followed by an event with activity B, and $|B > A|$ is the count of occurrences in the log where an event with activity B is followed by an event with activity A. This dependency threshold is comprised between -1 and 1. Activities A and B are considered in *dependency* if $|A => B|$ exceeds a dependency threshold. If two activities A and B are in dependency, then in the resulting process model, there is an edge connecting activity A and activity B.

III. METHOD

Applying the Heuristics Miner process discovery algorithm only to a small subset of the event log, the output might comprise:

- Some activities that are in clear dependency (e.g., the dependency value is very near to 1).
- Some activities that are not in dependency (e.g., the dependency value is below 0).
- Some activities that may be in dependency (might be slightly above or slightly below the dependency threshold).

The question is if the output of the application of Heuristics Miner to the small subset of the log is reliable; that means if the activities are in clear dependency in the small subset, then they are in dependency on the entire event log, and if the activities are not in dependency on the sample, then they are not in dependency on the entire log. We have to examine the following questions:

1) Is the subset of the log a representative sample of the entire log?
2) Is the subset of the log big enough to infer the dependencies?

To avoid taking an unrepresentative sample, random traces in the log should be taken. Indeed, taking the first traces in the log (e.g., the traces that happened first) could be dangerous as there could be a concept drift in the process [15][16]. This could be granted using a random-access storage like Apache HBase [17]. In order to reply to the second question, we have to do some assumptions on the probabilistic distribution of dependency values. If the dependency values calculated on various random-taken subsets of the log follow a normal distribution, then we can define a *confidence interval* on the dependency value between activities A and B, given a sample of size N (in the following formula, we denote with p the value $dep_{sample}(A,B)$), as:

$$dep_{entire}(A,B) \in \left(p - k\sqrt{\frac{p(1-p)}{N}}, p + k\sqrt{\frac{p(1-p)}{N}} \right)$$

Where k is a constant given by the chosen confidence (for example, $k = 1.96$ if we want a 95% confidence). The formula means that the value of dependency on the entire log $dep_{entire}(A,B)$ is comprised (with a confidence given by the value of k) in some interval centered on $dep_{sample}(A,B)$. For $N \to \infty$, e.g., for a big sample size, we can see that the interval length goes to 0 (this means that with a big sample size we get a dependency value that is equal or almost equal to the dependency value measured on the entire

 ISBN: 978-1-61208-542-5

log). However, even for a smaller sample, if $dep_{sample}(A, B)$ exceeds by much the dependency threshold, we could be quite sure that also $dep_{entire}(A, B)$ exceeds the dependency threshold, so a small subset of the log is enough to say that A and B are in dependency. Our proposal to determine a good sample size is described in the following algorithm that starts with an empty sample. The algorithm adds iteratively N traces (that are chosen randomly from the log) to the sample until a stop condition is reached; in each step, dependency measures between activities are calculated. The parameters of the algorithm are: N that is the number of traces added in each iteration; p that is described in the previous paragraphs; q that is the probability of doing another iteration of the algorithm.

1) Add N random traces to the sample.
2) Calculate the dependency values (in the sample) between activities.
3) For all activities that are in dependency on the sample, check the value $inf = p - k\sqrt{\frac{p(1-p)}{N}}$.
4) If for all activities that are in dependency on the sample, the value inf is above the dependency threshold, then return the dependency set found on the sample.
5) If there is no couple of activities where the value inf is above the dependency threshold, then do another iteration of the algorithm.
6) Otherwise, do another iteration of the algorithm with probability q and return the dependency set found on the sample with probability $1 - q$.

The output of the previous algorithm is a sample whose size is a multiple of N: if we do m iterations of the algorithm, then the (final) sample size will be mN. The proposed algorithm is probabilistic, as the produced sample is dependent on the traces that are chosen during its execution.

IV. Results and conclusion

The main assumption we have done on dependencies is that they follow a normal distribution, chosen a random sample of size N. We have checked this assumption (using Kolmogorov-Smirnov) on two event logs: "Road Traffic Fine Management Process" [18] (that contains 150370 traces) and "Receipt phase of an environmental permit application process" [19] (that contains 1434 traces). The process underlying these logs is very regular (it is a "lasagna" process); while the normality of dependencies in less regular, "spaghetti", processes (as "BPI Challenge 2015 Municipality 1" log [20]) is not equally clear.

There are several possible evaluation metrics of the sampling method proposed in the previous section which are based on several executions:

- (*E1*) Average (final) sample size as a fraction of the entire event log. As example, if we have a log with 1000 traces, choose $N = 100$ and, after the execution of the algorithm, we get a sample of size $4 * N = 400$, then $E1 = 0.4$.
- (*E2*) Average percentage of dependencies on the entire log that are dependencies also on the sample.
- (*E3*) Average percentage of dependencies in the sample that are not dependencies on the entire log.

We can propose an evaluation of the algorithm on the "Road Traffic Fine Management Process" and on the "Receipt phase of an environmental permit application process" logs. We have chosen 0.9 as dependency threshold, $k = 1.96$ for the confidence interval and $q = 1.0$ as the probability of doing another iteration of the algorithm. Moreover, we have chosen $N = 1000$ as sample size for "Road Traffic Fine Management Process" and $N = 10$ as sample size for "Receipt phase of an environmental permit application process". As the algorithm is probabilistic (sample size is dependent on which traces are chosen), it has been repeated on both logs for $M = 1000$ trials (Monte Carlo algorithm) averaging the metrics values recorded during the M executions.

For the "Road Traffic Fine Management Process" we have obtained the following values in the average of the proposed metrics: $E1 = 0.024$, $E2 = 0.9540$, $E3 = 0.6154$. For the "Receipt phase of an environmental permit application process" we have obtained the following values in the proposed metrics: $E1 = 0.0200$, $E2 = 0.9827$, $E3 = 0.2468$. So in both cases we can extract averagely over 95 % of the dependencies with less than 3 % of the entire log. Evaluation metric *E3* shows less good values, and some dependencies extracted on the sample are not dependencies on the entire log.

The point behind this paper is that, even if Big Data technology is becoming cheaper, unleashing such power to analyze an event log may not be useful as a small sample could still contain the dependencies we would find on the entire log.

References

[1] W. M. Van der Aalst and A. Weijters, "Process mining: a research agenda," *Computers in industry*, vol. 53, no. 3, pp. 231–244, 2004.

[2] W. M. van der Aalst et al., "Business process mining: An industrial application," *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[3] W. Van Der Aalst et al., "Process mining manifesto," in *International Conference on Business Process Management*. Springer, pp. 169–194, 2011.

[4] J. E. Cook and A. L. Wolf, "Automating process discovery through event-data analysis," in *Proceedings of the 17th international conference on Software engineering*. ACM, pp. 73–82, 1995.

[5] W. M. Van der Aalst and A. K. A. de Medeiros, "Process mining and security: Detecting anomalous process executions and checking process conformance," *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 3–21, 2005.

[6] W. M. Van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," *Information Systems*, vol. 36, no. 2 pp. 450–475, 2011.

[7] A. Vera-Baquero, R. Colomo-Palacios, and O. Molloy, "Business process analytics using a big data approach," *IT Professional*, vol. 15, no. 6, pp. 29–35, 2013.

[8] W. M. Van Der Aalst, "Decomposing process mining problems using passages," in *International Conference on Application and Theory of Petri Nets and Concurrency*. Springer, pp. 72–91, 2012.

[9] W. van der Aalst, "Process mining in the large," in *Process Mining*. Springer, pp. 353–385, 2016.

[10] J. Zhou, K.-M. Yu, and B.-C. Wu, "Parallel frequent patterns mining algorithm on gpu," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. IEEE, pp. 435–440, 2010.

[11] H. Reguieg, F. Toumani, H. R. Motahari-Nezhad, and B. Benatallah, "Using mapreduce to scale events correlation discovery for business processes mining," in *International Conference on Business Process Management*. Springer, pp. 279–284, 2012.

[12] J. Evermann, J.-R. Rehse, and P. Fettke, "Process discovery from event stream data in the cloud–a scalable, distributed implementation of the flexible heuristics miner on the amazon kinesis cloud infrastructure.", 2016

 ISBN: 978-1-61208-542-5

[13] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[14] A. Burattin, "Heuristics miner for time interval," in *Process Mining Techniques in Business Environments*. Springer, pp. 85–95, 2015.

[15] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *International Conference on Advanced Information Systems Engineering*. Springer, pp. 391–405, 2011.

[16] J. Carmona and R. Gavalda, "Online techniques for dealing with concept drift in process mining," in *International Symposium on Intelligent Data Analysis*. Springer, pp. 90–102, 2012.

[17] M. N. Vora, "Hadoop-hbase for large-scale data," in *Computer science and network technology (ICCSNT), 2011 international conference on*, vol. 1. IEEE, pp. 601–605, 2011.

[18] de Leoni, Mannhardt, "Road Traffic Fine Management Process". Eindhoven University of Technology. Dataset. http://dx.doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5, 2005

[19] Buijs, J.C.A.M., "Receipt phase of an environmental permit application process (WABO)", CoSeLoG project. Eindhoven University of Technology. Dataset. http://dx.doi.org/10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6, 2014

[20] B.F. van Dongen, "BPI Challenge 2015 Municipality 1", Eindhoven University of Technology. Dataset. http://dx.doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17, 2015

 ISBN: 978-1-61208-542-5