Evaluating the Quality of Discovered Process Models

A. Rozinat[1,2], M. Veloso[2], and W.M.P. van der Aalst[1]

[1] Information Systems Group, Eindhoven University of Technology, NL-5600 MB, Eindhoven, The Netherlands. {a.rozinat,w.m.p.v.d.aalst}@tue.nl
[2] Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213-3890, USA. veloso@cmu.edu

Abstract. In the domain of *process mining* the evaluation of models (i.e., "How can we measure the quality of a mined process model?") is still subject to ongoing research. Because the types of models used in process mining are typically on a higher level of abstraction (they, for example, allow to capture concurrency), the problem of model evaluation is challenging. In this paper, we elaborate on the problem of process model evaluation, and we evaluate both new and existing *fitness* metrics for different levels of noise. The new metrics and the noise generation are based on Hidden Markov Models (HMMs).

1 Introduction

Process mining deals with the discovery of *process models* (i.e., structures that model behavior) from event-based data. The goal is to construct a process model which reflects the behavior that has been observed in some kind of *event log*. An event log is a set of finite event sequences, whereas each event sequence corresponds to one particular materialization of the process. Process modeling languages, such as Petri nets [5], can then be used to capture the causal relationships of the steps, or activities, in the process. While many different process mining approaches have been proposed over the last decade, no standard measure is available to evaluate the quality of such a learned model [10]. Quality measures are needed because a learned model cannot always explain all the data, and there are multiple models for the same data ("Which one is the best?"). These problems are due to the fact that a log typically does not contain negative examples and that there may be syntactically different models having the same (or very similar) behavior. This paper deals with the topic of evaluating process models and we use Petri nets as a typical representation of the class of graph-based process modeling languages (EPCs, BPMN, UML Activity Diagrams etc.).

The remainder of the paper is organized as follows. First, we explain the problem domain of process model evaluation in more detail (Section 2). Then, we briefly sketch our approach (Section 3) and present experimental results (Section 4). Finally, we conclude the paper (Section 5).

2 Process Model Evaluation

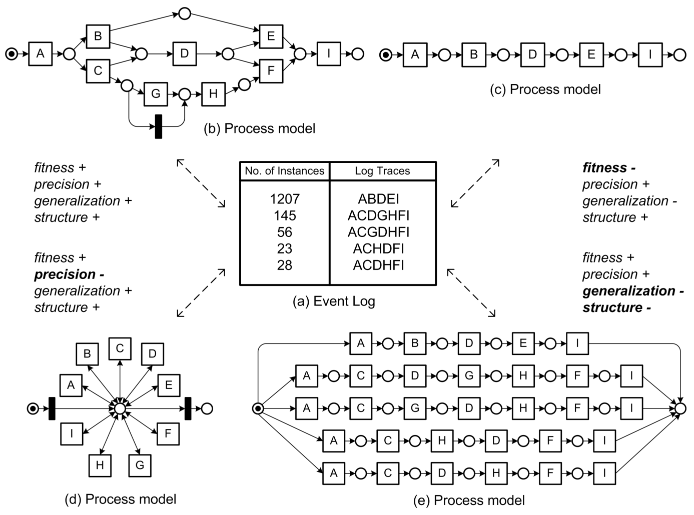
Process mining techniques focus on discovering behavioral aspects from log data. Since the mid-nineties several groups have been concentrating on the discovery of *process models* from event-based data. Process models are structures—usually directed graphs—that model behavior. The idea of applying process mining in the context of workflow management was first introduced by Agrawal et al. in [2]. Over the last decade many process mining approaches have been proposed. In [1] van der Aalst et al. provide an overview about the early work in this domain. While all these approaches aim at the discovery of a "good" process model, often targeting particular challenges (e.g., the mining of loops, duplicate tasks, or in the presence of noise), they have their limitations and many different event logs and quality measurements are used. Hence, no commonly agreed upon measure is available.

In [10] we motivate the need for a concrete framework that evaluates the quality of a process model, and thus enables (a) process mining researchers to compare the performance of their algorithms, and (b) end users to estimate the validity of their process mining results ("How much does this model reflect reality?"), and to choose between alternative models ("Which model is the best"?). It is important to understand that there is never only one single model for a given event log, but multiple models are possible due to two main reasons.

1. *There are syntactically different models having the same (or very similar) behavior.* Furthermore, there are numerous different modeling formalisms that can be employed. We do not focus on this aspect in our work.
2. *For a given input, an infinite number of models can be constructed.* Resulting models might not always accommodate all the traces in the event log, and they might allow for behavior not represented by any trace in the log.

To illustrate the second aspect, consider Figure 1 which depicts four different process models that could be constructed based on the event log in the center. The event log contains five different traces with frequencies ranging from 1207 to 23 instances per trace. For example, the sequence *ABDEI* (i.e., *A* followed by *B*, etc.) occurred 1207 times. While the model in Figure 1(c) does not accommodate all these five traces but only the first one (lack of *fitness*), the model in Figure 1(d) allows for much more behavior than just the traces in the event log (lack of *precision*).

Cook and Wolf [3] approach the discovery of Finite State Machine (FSM) models for software processes as a grammar inference problem, and, reflecting on the "goodness" of a model, they cite Gold [6] who showed that both *positive and negative* samples are required to construct 'accurate' (the FSM accepting all legal sentences and rejecting all illegal sentences of the language) and 'minimal' (the FSM containing the minimum number of states necessary) models. Furthermore, the samples must be *complete* (i.e., cover all possible inputs). However, the event logs used for process discovery cannot be assumed to be complete, and they normally do not contain negative examples. Note that the five traces in the event log in Figure 1(a) are positive examples, but no negative, or forbidden,



No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

Fig. 1. Process model evaluation can place in different dimensions [10].

traces are given. Furthermore, in some situations it can be the case that the positive examples are in fact distorted (noise), or contain exceptions that should not be included in the model. Therefore, process discovery algorithms have to face the following problems.

Dealing with Incompleteness If the log would be complete, it would be easy to assume that every sequence not present in the log is a negative example, and thus should not be possible according to the discovered model. Unfortunately, total completeness is an unrealistic assumption as the number of possible interleavings of concurrent activities increases exponentially[3]. Thus, *generalization* beyond the observed sequences to accommodate concurrent or combined behavior is often desirable.

Further Abstraction Besides generalizing to deal with incompleteness, further abstraction may be necessary to obtain meaningful process models. For example, in the presence of overly complex processes it often does not make sense to show a very detailed ("spaghetti-like") model. Furthermore, one might want to deal with noise, or show the main flow of a process and thus ignore possible exceptions. In theses case, abstraction can lead to models with a decreased *precision* and a decreased *fitness*.

So, we can see that—while on the first glance it seems logical to aim at models with perfect fitness and precision—this is not always desirable. Instead, algorithms strive to find "the right degree of abstraction", depending on the assumed circumstances and the purpose of the discovered model. As a consequence, process model evaluation needs to take these goals into account, and

[3] Already 5 concurrent activities can generate $5! = 120$ possible traces, and 10 concurrent activities can result in $10! = 3628800$ differently ordered sequences.

may have an unwanted bias if applied in the wrong context. For example, the model depicted in Figure 1(c) might be considered a good abstraction of the 80% most frequent behavior, but would be a relatively poor model if the goal is to obtain a complete picture of the overall process.

3 Approach

Although it is vital to develop practical methods that are able to take the desired abstractions made by process discovery algorithms into account when evaluating the resulting models, in reality there are often also simpler processes (or parts of processes) that only exhibit sequential routing, alternative behavior, and loops (but no parallelism). Examples of such processes can be found in administrative procedures employed in municipalities, insurance companies etc.

In our approach we focus on such simple processes with the assumption that the models should be as accurate, i.e., *fitting* and *precise*, as possible for a given event log. We define new evaluation metrics and compare them to existing evaluation methods used in the process mining field. It is important to note that here we do not seek for the all-encompassing standard measure (for this, posing restrictions on concurrency would not be a good idea), but rather aim at providing efficient base line metrics against which other evaluation metrics can be compared to see how they perform in these simple situations. Ultimately, the goal should be to give better support for what is the "right" quality metric for the situation at hand.

In our approach we use Hidden Markov Models (HMMs) [8] to represent these simpler models. HMMs, as opposed to plain Markov chains, allow for a separation of states and observation elements (the states are *hidden*). Then we use these HMMs to calculate metrics and generate logs with varying, yet determined levels of noise:

Metrics We define a mapping from Labeled Petri nets without parallelism onto HMMs, whereas we create one state per labeled task in the process model (unlabeled tasks are not represented by a separate state in the HMM as they are not observable), and where we first link the corresponding observation element with 100% probability to that state (but multiple states may be linked to the same observation). Based on this HMM and the event log, we define quality metrics for the fitness and precision dimension.

Noise Generation Then, we introduce successive degrees of noise into the HMM model (equally distributed over all states). We then use this HMM to simulate logs for different levels of noise, and evaluate the development of a number of fitness metrics over these logs with varying degrees of noise.

For more details, we refer the interested reader to our technical report [11], which in addition provides definitions of the used formalisms and our quality metrics, explains our mapping from Sequential Petri nets to HMMs based on a simple example, describes the precise experimental setup, and provides more detailed results. Furthermore, [11] also shows the differences in representational power by comparing HMMs and Petri nets as a modeling technique.

4 Experimental Results

We performed experiments for varying degrees of noise based on process models from different domains. First, we used four process models that were mined based on log data collected by the CMDragons team during the international robot soccer competition 'RoboCup' 2007, and thus constitute models of the behavior in a multi-agent robotic system. Second, we evaluated three different models of administrative processes within a municipality in the Netherlands. Finally, we selected three suitable models from the SAP reference models, which is a publically available model that contains more than 600 enterprise models, and analyzed them with our approach.

The original models were given in terms of Heuristics nets and EPCs, and they were translated into Petri nets using conversion facilities in the ProM framework, respectively. We performed experiments on these models with varying parameters. The results of these experiments are very similar, and in the following we use a single, but representative, example to point out the main conclusions that we can draw from them. The detailed results are provided in our technical report [11].

As for the process model evaluation metrics, we used the following two metrics from the process mining domain.

- The token Fitness [9] is based on replaying the log in a Petri net process model and relating the number of "missing" and "remaining" tokens during log replay to the total number of produced and consumed tokens (here referred to as *Token Based*). A similar metric is the Continuous Parsing Measure [12] which measures the number of missing and remaining activations while replaying a Heuristics net.
- The *Improved Continuous* semantics fitness [4] is used by the Genetic Miner to select the best process models in each generation of the genetic algorithm, and incorporates a fitness evaluation similar to the Continuous Parsing Measure, a precision evaluation, and gives some extra weight based on the number of traces that have problems.

Furthermore, we used the following three new HMM-based metrics, which are defined in our technical report [11].

- A *Trace Based* metric simply calculates the percentage of traces that have no errors given the model. Similar Metrics are used in the process mining domain (e.g., [12]).
- Another metric measures fitness on the *Model Level*, i.e., relating to how many "forbidden" transitions in the model have been "broken" by the log.
- The *Event Level* metric evaluates the occurrence of these "forbidden transitions" with respect to the whole log.

Now consider Figure 2, which depicts the $2+3=5$ fitness values (y axis) for 50 different noise levels (x axis), with 100 traces per log, and a maximum of 100 events per trace, for one of the models mined from the robot soccer data. Since

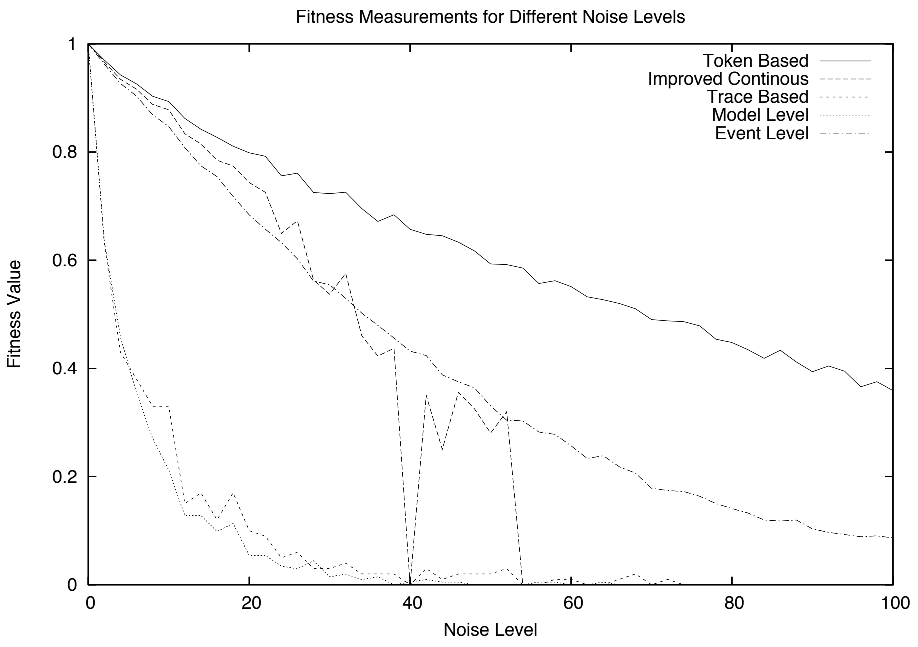


Fig. 2. Fitness values for 50 different noise levels on larger model.

this figure is representative for the larger set of experiments in [11], we can use it to illustrate our main conclusions from the experiments.

1. *Existing metrics have a bias when applied to simple models.* We can see that the Token Based fitness, which is representative for other similar fitness approaches in the process mining domain, does not drop to much less than a fitness of 0.4 throughout the whole experiment. This can be explained by the fact that the log replay "leaves tokens behind" for potential later use, which is appropriate for models containing parallelism (as one needs to look more than one step forward to satisfy all the dependencies) but not for simple Petri nets as evaluated in our experiments. Thus, in these situations, and more severely with an increasing level of noise, this metric provides overly optimistic results for sequential models.
2. *Metrics should measure only one thing.* We can see that the *Improved Continuous* fitness suddenly drops dramatically when the noise level is increased above 40%. This can be explained by the fact that this metric was designed to steer a genetic algorithm based on a mixture of different measurements. For that purpose, it is a good metric. However, if one wants to use the metric to gain insight into the quality of a process model, it becomes difficult as the interpretation of results becomes difficult ("Which aspect contributed to this value?", "Is it the property I want to measure?").
3. *Trace-based metrics do not make much sense for large models and long process instances.* Based on the *Trace Based* fitness, we can see that this metric

drops very quickly towards 0, already for low levels of noise. This is due to the fact that in case of longer process instances (as here up to 100 events per trace) already one small error in the trace renders the whole trace to have a negative impact on the overall fitness. With an increasing level of noise, there will be soon no more traces that have no errors, thus rendering the measure to provide pessimistic results if compared to our notion of noise.

The results indicate that the *Event Level* metric seems to be the best metrics in our setup. This is not surprising since it best matches the notion of noise used in the experiments. If we would, for example, generate noise by distorting an increasing number of log traces, the *Trace Based* metric would match this notion best. Furthermore, one could introduce distortions gradually over different parts of the process, which would render the *Model Level* metric less pessimistic.

Note further that, depending on the goal of the evaluation, the evaluation setup can be different. For example, we here assumed that it is desirable for a fitness metric to scale as linearly as possible with respect to the degree of distortion in the log. Since lower portions of noise (e.g., 5% or 10%) are much more common in real processes than a very high portion (e.g., 80%), one might prefer the metric to scale linearly from, e.g., 0% (metric yields 1) to 20% noise (metric yields 0). However, for a first general evaluation the presented setup seems reasonable.

5 Discussion

We generally use metrics to obtain information about the quality of a process model. But to be sure that the conclusions that we draw from the measurements are valid, we must first ensure the *quality of the metric itself*. The following requirements are are generally considered relevant [7] to ensure the usefulness of a metric: *validity* (the measure and the property to measure must be sufficiently correlated), *reproducibility* (be independent of subjective influence), *stability* (be as little as possible affected by properties that are *not* measured) and *analyzability* (relates to the properties of the measured values).

Given the fact that the validity and reproducibility are typically not an issue, we have proposed a structured approach to evaluate the analyzability and stability of certain process mining metrics. As a first step, we focused on simple process models and the fitness and precision quality dimensions. We have seen that—in this simple setting—existing process mining metrics can yield overly optimistic results, which affects their *analyzability*. Furthermore, some metrics measure more than one aspect, which makes interpretation of the results difficult and negatively affects their *stability*. Further research is required to develop evaluation approaches for further dimensions and more complex scenarios. The ultimate vision for process model evaluation would then be to have a methodology that assists in selecting the “right” metric for the “right” situation, and based on the goal of the evaluation.

Acknowledgements

This research is supported by the IOP program of the Dutch Ministry of Economic Affairs. The authors also thank the CMDragons team for sharing their RoboCup log data, and particularly Stefan Zickler and Douglas Vail who helped to create the ASCII logs used for conversion to the MXML format. Furthermore, we thank the town hall in the Netherlands for letting us use their process models, and Florian Gottschalk and Boudewijn van Dongen for the EPC reference models. Finally, we thank all ProM developers for their on-going work on process mining techniques. Special thanks to Christian W. Günther for NikeFS2, which makes it possible to work with large real-world logs, and for his UI redesign, which continues to impress people and makes using ProM so much more fun.

References

1. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
2. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
3. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
4. A.K. Alves de Medeiros. *Genetic Process Mining*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2006.
5. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
6. E.M. Gold. Complexity of Automaton Identification from Given Data. *Information and Control*, 37(3):302–320, 1978.
7. P. Liggesmeyer. *Software-Qualität – Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2002.
8. L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
9. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
10. A. Rozinat, A.K. Alves de Medeiros, C.W. Günther, A.J.M.M. Weijters, and W.M.P. van der Aalst. The Need for a Process Mining Evaluation Framework in Research and Practice. In Arthur H. M. ter Hofstede, Boualem Benatallah, and Hye-Young Paik, editors, *Business Process Management Workshops*, volume 4928 of *Lecture Notes in Computer Science*. Springer, 2008.
11. A. Rozinat, M. Veloso, and W.M.P. van der Aalst. Using Hidden Markov Models to Evaluate the Quality of Discovered Process Models. Extended Version. BPM Center Report BPM-08-10, BPMcenter.org, 2008.
12. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.