Incremental Mining of Predictive Process Models based on Declarative Formalism

Sergio Angelastro

1st year PhD Student
Dipartimento di Informatica, Università di Bari, Bari, Italy
sergio.angelastro@uniba.it
Advisor: Stefano Ferilli

Abstract. Workflow management is fundamental to efficiently, effectively and economically carry out complex processes in domains such as the industrial one or others related to human behavior. The need is to understand and formalize these processes in models that can support their analysis, replication and enforcement. These are complex, costly and error prone to be built manually. Hence, the interest in automatically learning them (preferably incrementally in real time settings) from examples of actual procedures. It is also desirable to be able to provide useful operational support. One of the aims of my PhD is to exploit and extend Process Mining methodologies in order to develop Declarative Process Models that are able to make various kinds of predictions.

1 Introduction

Contemporary society is pervaded by procedures characterized by complex interactions of different inter-related tasks. Proper definition, handling and management of such interactions are crucial for (effectively and efficiently) reaching the given objectives. This requires the development of specific techniques and tools to model workflows and to enact them. The availability of a suitable workflow model can determine the production and economic success of a company, or user satisfaction in a Smart Home Environment. While most information for setting up these models comes from the practitioners, that day by day carry out and improve the procedures, formal models of activities are typically produced by supervisors and managers. This gap often causes models not to perfectly fit the practice. Also, producing the models is inherently complex, costly and error-prone [7]. A related problem is the need for adapting and fine-tuning existing models in dynamic environments. These are strong motivations for developing techniques to automatically learn, manage and refine workflows, which are some of the tasks of Process Mining [14, 8]. A classical application domain is the industrial one, where the activities of a production process must be monitored and checked for compliance with a desired behavior. The availability of a formal model of the desired behavior can automate the supervision task. Given an intermediate status of a process execution and some knowledge about how the execution will proceed might allow the (human or automatic) supervisor to take

suitable actions to support the next activities. In an industial environment the rules that determine how the process must be carried out are usually quite strict. So, the emphasis is more on conformance checking, while prediction of process evolution is more trivial. On the other hand, in other, less traditional application domains for process managament, that have been introduced more recently (e.g., the daily routines of people, at home or at work), much more variability is involved, and obtaining reliable predictions becomes both more complicated and more useful.

My research focuses on a novel framework for process mining and management, based on First-Order Logic (FOL for short). It lies in the Declarative Process Mining setting, a perspective that has been recognized to be very important when dealing with particularly complex models and domains [9]. Incremental learning of process models, expressiveness of the representation and efficiency are the most outstanding features of this framework. An objective of my research is improving the predictiveness of the process models, so that at any moment during the process execution they are able to suggest which activities will be carried out next. This objective is being pursued using two different approaches, a heuristic one and a bayesian one. The former has already been implemented and tested, and performed better than the current state-of-the-art. For the latter the experimental phase is currently starting.

2 Background and Related Works

A *process* consists of actions performed by agents (humans or artifacts) [1, 3], which are formally specified in a *workflow* that express how they can be composed to result in a valid process. Allowed compositional schemes include sequential, parallel, conditional, or iterative execution [11]. A process execution can be described in terms of *events*, i.e. identifiable, instantaneous actions (including decisions upon the next activity to be performed). A *case* is a particular execution of actions compliant to a given workflow. Case *traces* consist of lists of events associated to *steps* (time points) [12]. A *task* is a generic piece of work, defined to be executed for many cases of the same type. An *activity* is the actual execution of a task by a *resource* (an agent that can carry it out). Relevant events are the start and end of process executions, or of activities [3].

According to [1, 7], the framework input formalism consisting of 7-tuples

$$\langle T, E, W, P, A, O, R \rangle$$

where T is the event timestamp, E is the type of the event (one of 'begin_process', 'end_process', 'begin_activity', 'end_activity'), W is the name of the reference workflow, P is the case identifier, A is the name of the activity, O is the progressive number of occurrence of that activity in that case, and R (optional) specifies the resource that is carrying out the activity. Compared to other settings in the literature, this formalism allows to explicitly express parallelism among activities [5]. As regards the output formalism, the structure of a process model is described using the following elements:

tasks : the kinds of activities that are allowed in the process;
transitions : the allowed connections between activities, in the form $t : I \Rightarrow O$

where I and O are multisets of tasks. Transitions carry the information about the flow of activities during process execution. A transition t is enabled if all input tasks in I are active; it occurs when, after stopping (in any order) the concurrent execution of all tasks in I, the concurrent execution of all output tasks in O is started (again, in any order). For analogy with the notions of 'token' and 'marking' in Petri Nets, during a process enactment we call a token an activity that has terminated and can be used to fire a transition, and a marking the set of current tokens.

The activity prediction task has received very little attention in literature. In [13] it is cast as a recommendation problem. The process model is used to provide a ranked list of the activities currently enabled, given the partial execution of the enacted process and historical information. The experimental results reported in [13] show that the more historical information used, the better the quality of the recommendation. However, the approach considers traces that are simple sequences of activities, contrary to our approach. A bayesian approach was used in [3] to guess the next activity in a process case, as a function of its frequency. Starting from a Markov approach to infer a formal model, the authors move towards a Bayesian problem to infer ativities. However, this approach was not thoroughly investigated. Some approaches naturally fit very well-structured process, but they suffer problems related to incompleteness, noise, underfitting and overfitting. The approach considered in [2] overcomes these problems using customized sequential pattern mining algorithm. Unfortunately, concurrent behavior is not handled.

3 Prediction Methodology

The problem of prediction is set as follows: given a model and a partial status of a process during its execution, the output is a ranked list of possible activities to be carry out next. Both prediction approaches I am developing, heuristic and bayesian, try to foresee the next activity based on the information produced during the conformance checking of a case during its enactment. The task of conformance checking is in charge of evaluating the alignment of a new case with a given model. However, as shown in [6], given an intermediate status of the process enactment and a new activity that is started, there may be different alignments that are compliant with the new activity, and one may not know which is the correct one until a later time. When the next event is considered, each corresponding evolution of the current alignments is a possible status of the process enactment that the must be carried on by the system. New events may point out that some current alternate statuses were wrong. So, as long as the process enactment proceeds, the set of alternate statuses that are compliant with the activities carried out so far can be both expanded with new branches, and pruned of all incompatible alternatives with the activities carried out so far. Each status is represented by a 5-tuple

$$\langle Marking, Ready, Case, Transition, Warning \rangle$$

where $Marking$ is the set of terminated activities not yet used to fire a transition, *Ready* is the set of output activities of fired transitions (expected to be carried out next), $Case$ is the multiset of compliant historical cases (it is a multiset because a task can be executed many times in a single case), $Transition$ is the multiset of fired transitions to reach that status (it is a multiset because a transition can be applied many times) and $Warning$ is the multiset of raised warning (it is a multiset because the same warning can appear many times).

Concerning the heuristic approach, information about alignment is used in the ranking strategy. Specifically, the approach exploits information concerning both the support frequency of $Case$ and the warnings about incompliant case features (e.g., unexpected task or transition, preconditions not fulfilled, unexpected resource running a given activity, etc.) raised by the compliance check. Warnings are associated to weights that quantify their degree of severity. Activities in the *Ready* sets of compliant statuses are scored and ranked based on a heuristic combination of the following parameters:

$\mu(a)$: multiplicity of a (ready activity) across various statuses;
C_{status} : support frequency of a status;
$\delta(status)$: sum of weights of warnings raised by the status in which a is included

The basic idea underlying the combination is that an activity a is more likely to be carried out next if it appears in more statuses, which have a higher support frequency and a lower sum of warning weights.

As regards the other approach, a bayesian model is trained on the historical cases. For each known/observed task a, the frequencies of 6 kinds of features are incrementally learned for each process event, as long as the compliance check proceeds:

$$F_a = \langle Run\_task_a, End\_task_a, Mark\_task_a, Run\_trans_a, End\_trans_a, Mark\_trans_a \rangle$$

where Run_task_a refers to the set of tasks that are running when a is started, End_task_a refers to the set of tasks ended before a was started, $Mark\_task_a$ refers to the set of tasks ended before a was started and not yet consumed, Run_trans_a refers to the set of transitions that are running when a was started, End_trans_a refers to the set of transitions ended before a was started, and $Mark\_task_a$ refers to the set of fired transitions still having marked output tasks when a was started. During the supervision process of a new case, for each status, the information about the 6 features of the current case is maintained, so that each ready activity a can be scored and ranked based on its probability conditioned by its features F_a:

$$P(a|F_a) = \frac{P(a) \cdot P(F_a|a)}{p(F)}$$

where $P(a)$ is the prior probability of a, $P(F)$ is the prior probabilty of the features, and $P(F_a|a)$ is the probability that the features are valid when a is started.

The expected advantage of this approach is that, differently of the heuristic one, it can provide hints even when the supervision is no longer able to find valid alignments between the current execution and model. Indeed, since it exploits both the workflow model and the bayesian model of the process, it is more likely that the robustness of the operational support is guaranteed.

4 Applications in real domains

The learning task based on the hybrid approach has been tested on different datasets collected in smart home environments. The Aruba dataset, from the CASAS repository [10], concerns daily activities of people living in cities all over the world and contains a set of annotated high-level activities associated to the log of outputs obtained along time by several sensors placed in the surveyed home environment. We also used another dataset, GPItaly, built by extracting the data from one of the Italian use cases of the GiraffPlus project [4]. It concerns the movements of the house's inhabitants, instead of their activities, in order to learn typical paths.

Experimental results show that the approach is efficient and effective for learning and modeling daily routines of people, since in general there is a convergence towards a stable model, requiring less and less model refinements, as long as more cases are processed. Activity prediction was made in more than 85% of the cases (when a reliability threshold is not passed, the approach abstains from the prediction), and in 97% of these cases the correct activity was present in the ranking, in which case it was always at the top [6]. These figures raise to 99% and 97% in the case of movements prediction [6].

5 Remarks and Ongoing Work

In addition to other classical exploitations, process models may be used to predict the next activities that will take place. This would allow to take suitable actions to help accomplishing those activities, during a process enactment. One of the interests of my PhD research is related to this problem. This extended abstract introduced two approaches (heuristic and bayesian) to make these kinds of predictions using Declarative Process Mining. Experimental results on different domains suggest that the hybrid approach can successfully perform such predictions.

Currently, experiments on the bayesian approach are about to start, in order assess its performance and compare it to the hybrid approach. The aim is to understand which one performs better and when, in order to identify a possible cooperation of the two. A test of the performance on other domains is needed, especially in those that exhibit more complex features, such as high parallelism between activities. Furthermore, an investigation of the quality of the models based on how good they are in supporting prediction is a relevant and interesting research direction, since it represents an open issue in Process Mining.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, 1998.
2. M. Ceci, P.F. Lanotte, F. Fumarola, D.P. Cavallo, and D. Malerba. Completion time and next activity prediction of processes using sequential pattern mining.
3. J.E. Cook and A.L. Wolf. Discovering models of software processes from event-based data. Technical Report CU-CS-819-96, Department of Computer Science, University of Colorado, 1996.
4. S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, J. Gonzalez, L. Karlsson, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo, F. Pecora, S. von Rump, Štimec, J. Ullberg, and B. tslund. Giraffplus: Combining social interaction and long term monitoring for promoting independent living. In *Proc. of the 6th International Conference on Human System Interaction (HSI)*, pages 578–585. IEEE, 2013.
5. S. Ferilli and F. Esposito. A logic framework for incremental learning of process models. *Fundamenta Informaticae*, 128:413–443, 2013.
6. S. Ferilli, F. Esposito, D. Redavid, and S. Angelastro. Predicting process behavior in woman. In *Proceedings of the XV International Conference of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence - Volume 10037*, AI*IA 2016, pages 308–320, New York, NY, USA, 2016. Springer-Verlag New York, Inc.
7. J. Herbst and D. Karagiannis. An inductive approach to the acquisition and adaptation of workflow models. In *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 52–57, 1999.
8. IEEE Task Force on Process Mining. Process mining manifesto. In *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. 2012.
9. M. Pesic and W. M. P. van der Aalst. A declarative approach for flexible business processes management. In *Proceedings of the 2006 international conference on Business Process Management Workshops*, BPM'06, pages 169–180. Springer-Verlag, 2006.
10. P. Rashidi and D. J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(5):949–959, 2009.
11. W.M.P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8:21–66, 1998.
12. W.M.P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16:1128–1142, 2004.
13. B. Weber, B. F. Van Dongen, M. Pesic, and C. W. Guenther. Supporting flexible processes through recommendations based on history.
14. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering workflow models from event-based data. In V. Hoste and G. De Pauw, editors, *Proceedings of the 11th Dutch-Belgian Conference of Machine Learning (Benelearn 2001)*, pages 93–100, 2001.