Improving Process Mining Prediction Results in Processes that Change over Time

Alessandro Berti
SIAV
35030 Rubano PD (Italy)
Email: alessandro.berti89@gmail.com

Abstract—In this paper, we propose a method in order to improve the accuracy of predictions, related to incomplete traces, in event logs that record changes in the underlying process. These “second-order dynamics” hamper the functioning of Process Mining discovery algorithms, but also hamper prediction results. The method is simple to implement as it is based exclusively on the Control Flow perspective and is computationally efficient. The approach has been validated on the Business Process Intelligence Challenge 2015’s Municipality 5 event log, that contains an interesting shift in the process due to the union of the municipality with another municipality.

Keywords—Concept Drift; Process Mining; Prediction.

I. INTRODUCTION

Business processes are constantly evolving to adapt to new opportunities, and continuous improvement is needed for a company in order to remain at the top. In some cases, the quality of a process can be measured in time: for example, in Service Desk tickets, avoiding to break service level agreements is important. Knowing in advance which instances are most critical, assigning more resources to them, may be vital for some organizations.

Here arises the need of a good prediction algorithm for process instances. Process Mining provides some techniques to predict the completion time of instances, however they assume the underlying process to be static, obtaining in many cases poor results. In this paper, we provide an approach to improve existing prediction results by considering the fact that the process changes over time. This is the first approach in the field. An assessment done on Business Process Intelligence Challenge 2015’s Municipality 5 event log shows that the approach actually improved the prediction results in a process that changed over time.

The rest of the paper is structured as follows. In Section 2, we present Process Mining techniques and a classification of concept drifts in processes. In Section 3, the method to consider concept drifts in the predictions is introduced. In Section 4, we show some results on the BPI Challenge 2015 log. We conclude in Section 5. The rest of the paper is structured as follows. In Section 2, we present Process Mining techniques and a classification of concept drifts in processes. In Section 3, the method to consider concept drifts in the predictions is introduced. In Section 4, we show some results on the BPI Challenge 2015 log. We conclude in Section 5.

II. BACKGROUND

Process Mining [1] is a relatively new discipline that aims to automatically discover and measure things about processes. It mainly uses automatic recordings of events which are event logs. Sub-disciplines of Process Mining are process discovery [2], that aims to automatically discover the process schema starting from an event log, process conformance [3] that is useful to see differences between a de-jure process model and the current executions of the process (recorded in the event log), process performance [2] that wants to identify bottlenecks inside business processes starting from event logs, and process-related predictions which will be analyzed later. Event logs are organised in traces that are collections of events serving to a particular purpose. For example, a trace might regard a single case served by an Help Desk process. Meanwhile, events can be described by several attributes, including:

- The activity that has been performed.
- The originator of the event (the organizational resource that has done the event).
- The timestamp (the time in which the event has been executed).
- The transition of the event that refers to the state of execution (a “complete” transition means that the activity actually ended, a “start” transition means that the activity started).

The trace itself can be characterised by several attributes (for example, in an Help Desk system, the severity of the case might be an attribute). The minimum timestamp of its events can be considered as start timestamp of the trace, and the maximum timestamp of its events as end timestamp. Many times, there is only a transition (“complete”), so the trace might be described (in the Control Flow perspective) by the succession/list of its activities. This is a condition required by some Process Discovery algorithms, like the Heuristics Miner [4] that aims to discover the process schema by calculating the dependency between activities. This means that if in all occurrences of an event log an activity (1) is followed by another activity (2), then Heuristics Miner can discover a process schema in which activity 1 is always followed by activity 2. So, Heuristics Miner analyzes (among the others) the paths in a trace: a path is a direct succession of activities in a trace. For example, if a trace contains (analyzing only the Control Flow perspective) the activities ABCDE then all the paths contained in the trace are: AB BC CD DE. A path belongs to a trace if it is contained in the trace. An important definition provided for later use is about the belonging of a trace to a time interval. A trace, with start as the start timestamp and end as the end timestamp, belongs to a time interval \([t_1, t_2]\), if one of the following three conditions is satisfied:

1) \(start \leq t_1 \leq t_2 \leq end\)
2) \(t_1 \leq start < t_2\)
DiffInt(log, I₁, I₂)

Require: An event log log, time sub-intervals I₁ and I₂.

\[ T_{I_1} = \{ tr \in log, tr \in I_1 \} \]
\[ T_{I_2} = \{ tr \in log, tr \in I_2 \} \]

RelImp₁ = \{ \{ path, \frac{\#occ. path}{\#Tr₁} \} | \exists tr \in T_{I_1}, path \in tr \}

RelImp₂ = \{ \{ path, \frac{\#occ. path}{\#Tr₂} \} | \exists tr \in T_{I_2}, path \in tr \}

AllPaths = \pi₀(RelImp₁) \cup \pi₀(RelImp₂)

\[ D = \emptyset \]

for \( P \in \text{AllPaths} \) do
  if \( P \in \pi₀(\text{RelImp}_1) \) and \( P \in \pi₀(\text{RelImp}_2) \) then
    \[ D[P] = \frac{\text{Abs(}\text{RelImp}_1[P], \text{RelImp}_2[P])}{\max(\text{RelImp}_1[P], \text{RelImp}_2[P])} \]
  else
    \[ D[P] = 1 \]
  end if
end for

return \( D \)

Figure 1. The algorithm to calculate the difference between the paths’ importance in two different sub-intervals.

Importance(tr, D)

Require: A complete trace \( tr \), difference of importance of paths between intervals \( D \).

return \[ \text{avg}_{(A₁, A₂) \in \text{Paths}(tr)} \{ 1 - D[(A₁, A₂)] \} \]

Figure 2. The algorithm to calculate the importance of a trace \( tr \) in the difference of temporal contexts described by the dictionary \( D \).

Similarity(log, tr₁, tr₂, intervals)

Require: An event log log, an incomplete trace \( tr₁ \), a complete trace \( tr₂ \) (used for prediction purposes), collection of time sub-intervals.

if \( \exists I \in \text{intervals}|tr₁ \in I, tr₂ \in I \) then
  return 1
end if

return \[ \max_{I₁, I₂ \in \text{intervals}|tr₁ \in I₁, tr₂ \in I₂} \text{Importance}(tr = tr₂, D = \text{DiffInt}(log, I₁, I₂)) \]

Figure 3. The algorithm to calculate the similarity between the temporal context of two traces, one of them is incomplete and the other is complete and used for prediction.

3) \( t₁ < \text{end} \leq t₂ \)

It might be important also to consider the difference between complete and incomplete traces. The last ones are being reported in the log, although their execution is not finished. The distinction is somewhat difficult to make, [5] can be referred for further discussion. A possible way to detect incomplete traces is applying heuristics to the end activities: if the end activity of a trace can be found as an end activity in many other traces, then it is considered to be a complete trace, otherwise incomplete. The succession of the activities of an incomplete trace might be shared with a complete trace, being a “prefix”. An interesting task about incomplete traces might be the prediction of their attributes. For previous work on prediction tasks, [6] that mainly describes a method for the prediction of the remaining time of incomplete traces. Basically, the idea is to build an annotated “transition system” (the explanation of this concept is skipped, as it is not firmly connected with the explanation of the method. For further discussion, see [6]), that is learned from previous executions, i.e., complete traces, using an abstraction mechanism. In [7] is proposed a method to predict the remaining time based on sequential pattern mining.

A further step is the one explained in [8]. The prediction of the remaining time is calculated using these two factors:

- The likelihood of following activities, given the data collected so far.
- The remaining time estimation given by a regression model built upon the data.

Basically, this method is an improvement over [6] because it considers not only the Control Flow perspective, but also other events’ attributes, identifying the ones that are relevant to the prediction of the remaining time. A process specialist could insert artificial attributes to events (for example, the workload of the resources, or the work in process), in order to improve the prediction. However, an aspect somewhat ignored in predictions is the fact that the underlying process might change during time. As [9] reports, changes might induce one of the following drifts:

- Recurring drifts: these ones refer to changes that happen in some moments of the year (seasonal influence) or some other recurring changes (for example, a financial process might change near the financial closure of the year).
• Sudden drifts: these refer to big changes in the process: the “old” process cease to exist, while a “new” process starts to be applied.
• Gradual drifts: these refer to a gradual shift from an “old” process to a “new” process. This might be done to let the organizational resources learn the new process.

A method to identify and to cope with changes in the process is described always in [9]: at a first time you have to identify change points in the process (i.e., the times when the process is different), after that you have to identify the region of the change and the type of the change (recurring, sudden, gradual drifts). The last step exploits this knowledge to “unravel” the evolution of the process, describing the change process. Basically, an application of the classical Process Discovery algorithms (for example Heuristics Miner [4], Inductive Miner [10]) can be reliable only in time intervals that contains a consistent, without-drifts process. The same is valid for the prediction algorithms, as a prediction based on the entire process (that might be changed meanwhile) is not-so-accurate. However, also a prediction based only on the last iteration of the process might be incomplete and not-so-accurate.

III. METHOD

The proposed method wants to overcome the limitations of both a prediction based on the entire process, and a prediction based only on the last iteration of the process (it might be a restricted time interval). A method to detect change points and analyze them is not proposed, for this scope, [9], [11] can be referred; the proposed method starts from the assumption to know where change points are (this could also be done with an interview to organizational resources). Starting from the overall time interval of events contained in an event log, it is supposed that there is a collection of time sub-intervals covering the entire time interval and in which the underlying process is constant.

The method is based on the knowledge of a distance measure between two time sub-intervals. This way, you have a method to say how much reliable a complete trace (that might be following a slightly different process) is in the prediction of an incomplete trace that is based on the last iteration of the process. The proposed algorithm in Figure 1 measures the distance path by path, as some paths might be equally present in both intervals. Algorithm in Figure 1 basically works calculating the relative importance of each path in each of the subintervals (that is the ratio of the number of path’s occurrences and the number of traces), and then comparing this quantity between the intervals. The reliability of the trace in the context of a prediction can be then calculated using the algorithm in Figure 2. It is proposed to use the average (done on all the paths of a trace) of the distance calculated using algorithm in Figure 1. Other statistics (like the maximum of the distance) proved to be less reliable.

Algorithm in Figure 3 uses the previous two algorithms, starting from a couple of traces (the first of them is the one to predict), the event log and the subdivision in sub-intervals. It tries to find two sub-intervals, containing respectively the two traces (with the meaning explained in Background) that are at a minimum distance, so maximising the similarity. This has been done in order to avoid giving unnecessary low weights of similarity to traces whose duration has been longer than the sub-intervals in which the underlying process is constant.

Then, to obtain the prediction, one could use van der Aalst’s [6] algorithm, weighting the traces used for the prediction through algorithm in Figure 3.

IV. RESULTS

The proposed algorithms have been tested on the BPI Challenge 2015’s Municipality 5 event log (DOI 10.4121/uuid:b32c6fe5-f212-4286-9774-58dd53511cf8). The log describes a very complex process, with many activities, and is particularly interesting because this municipality (Municipality 5) got merged with another municipality (Municipality 2, DOI 10.4121/uuid:63a8435a-077d-4ece-97cd-2c76d394d99c) at a certain point of time, and the process became different. Some different time intervals can be identified:

1) The first one, going from the start of the log to June 2013, in which Municipality 5 was substantially departed from Municipality 2.
2) The shift one, going from June 2013 to June 2014, in which Municipality 5 get merged with Municipality 2.
3) The second one, going from June 2014 to the end of the log, in which Municipality 5 is already united with Municipality 2.

These sub-intervals were identified with a resource analysis, seeing that the resources working in the process got more numerous, and the point of the shift is comprised between June 2013 and June 2014. Being these sub-intervals roughly identified, the shift interval will be ignored for prediction purposes, and the focus will be on the first and the second interval, in which the underlying process is different.

The algorithm proposed by van der Aalst [6] is used as prediction (of the remaining time) algorithm, weighting the traces used for the prediction using Algorithm 3. All the traces in the log have been considered as completed ones, so for the prediction purposes a prefix of each one has been taken, the completion time has been predicted and compared to the effective completion time. The effectiveness of the prediction was measured using two standard measures (Mean Absolute Percentage Error (MAPE) and Root of Mean Squared Percentage Error (RMSPE)), briefly explained below. Here, $A_i$ is relative to the actual value (the effective completion time) and $F_i$ is relative to the predicted completion time.

\[ MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{A_i - F_i}{A_i} \right| \]
\[ RMSPE = \sqrt{\frac{\sum_{i=1}^{n} (A_i - F_i)^2}{n}} \]

In Table I, there are some results of the application of the algorithm in Figure 1 to Municipality 5 event log. The first column describes the path, the second and the fourth report the count of the paths in the respective time intervals, the third and the fifth report the relative importance (the average of the occurrences of paths inside traces). The sixth column is then calculated as the ratio of the absolute difference of the relative importances and the maximum of the two relative importances. You can see that for some paths there is a big difference in importance between intervals. This reflects the big change in the underlying process.
In Table II, we present some results related to predictions. Three different conditions have been compared:

- The prediction (of the remaining time) relative to a prefix of a trace (belonging to the first or second time interval), using for the prediction all the traces in the log.

- The prediction relative to a prefix of a trace belonging to the first interval, using for the prediction all the traces weighted accordingly to the algorithm in Figure 3.

- The prediction relative to a prefix of a trace belonging to the second interval, using for the prediction all the traces weighted accordingly to the algorithm in Figure 3.

The prefix is formed by the first two activities. The results are then grouped based on their prefix.

In Table III the same techniques are applied to a prefix containing the first three activities of the trace. In many occurrences prediction results obtained by weighting the traces using algorithm in Figure 3 are improved in comparison to the classical technique.

V. CONCLUSION AND FUTURE WORK

In this paper is proposed a method to consider process drifts in the prediction of traces’ attributes. At best of the author’s knowledge, this is the first approach in the field (so there are not comparisons with other methods). The method assumes that the times in which the process changes are already
known. All these changes, might they be seasonal, gradual or sudden, split the overall time interval into subintervals in which is assumed that the process is constant. The discovery of these times could be done in an automated way, for example using the algorithm described in [9], or manually through an interview. For each time sub-interval, you can observe how many times two activities are in direct succession; after that, you could compare the distributions measured in the different sub-intervals. This is useful to understand how much the process is different between different sub-intervals. This is useful to understand how much the process is different between different sub-intervals, and to give a different weight to the different (complete) traces one could use to predict the outcome of an incomplete trace. This is useful in each type of prediction, as the prediction of the remaining time in a trace.

The described algorithms are pretty easy to implement, and are not computationally expensive (the implementation has been realised in a plain Python script). However, the approach considers only the control flow perspective, and ignores other perspectives (like the data perspective and the resource perspective) in which the process could change over time. Indeed, changing roles inside an organizational process might change the throughput times, because of different skills, changed workloads and difficulties in collaboration between different work groups. Some literature can be cited related to social and work psychology [12], [13], [14] that give insights on how much inter-group relationships are important for organizational performance. Generally, one could identify inter-group distances in a process by measuring times elapsing between activities performed by different roles. This can be related to the Lean Manufacturing concept of Flow Rate [15], [16], [17]. Another aspect is related to the group’s Transactive Memory [18], [19], [20]. Transactive Memory is a psychological concept that could be explained as “group memory” and is related to the specialization and the coordination of the group [21], [22]. Indeed, a change in the work group’s structure that could be motivated by a change in the process, can hamper a lot the group’s performance, because of the newcomers’ need to know the rest and the roles of the group, or some people exiting the group. It is a pity that Transactive Memory in groups is generally difficult to measure [23], because it’s a powerful tool to measure group performance.

There is also scope to research related to non-instantaneous events that could include several transitions (start, complete, stop, resume) [24], as the framework described here works only for instantaneous events (each trace could be described by a succession of conclusive activities). Overall, the proposed method seems to be good performing on the BPI Challenge’s Municipality 5 log. In that log, the process changes after the union with another municipality (Municipality 2). Not in every event log, however, a change in the underlying process can be observed. There is also scope to research related to non-instantaneous events that could include several transitions (start, complete, stop, resume) [24], as the framework described here works only for instantaneous events (each trace could be described by a succession of conclusive activities). Overall, the proposed method seems to be good performing on the BPI Challenge’s Municipality 5 log. In that log, the process changes after the union with another municipality (Municipality 2). Not in every event log, however, a change in the underlying process can be observed. In that case, the method is useless.

Moreover, current results related to prediction of attributes (e.g., remaining time) are not that good, even with the proposed improvement. There is something more to come in order to get good and reliable predictions.

REFERENCES


Copyright (c) IARIA, 2016. ISBN: 978-1-61208-510-4

41


