ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Process Mining in Non-Stationary Environments

Phil Weber*, Peter Tiňo and Behzad Bordbar

School of Computer Science - University of Birmingham - UK
Email: {p.weber,p.tino,b.bordbar}@cs.bham.ac.uk

Abstract. Process Mining uses event logs to discover and analyse business processes, typically assumed to be static. However as businesses adapt to change, processes can be expected to change. Since one application of process mining is ensuring conformance to prescribed processes or rules, timely detection of change is important. We consider process mining in such non-stationary environments and show that using a probabilistic view of processes, timely and confident detection of change is possible.

1 Introduction

Business processes describe related activities which are carried out to fulfil a business function. Fig.1 shows an example process, depicted as a Petri net and probabilistic automaton. As the process is executed, information will be recorded in log files. Abstracting from detail, the 'trace' of a single enactment of this process might be recorded as a string of symbols (activities). Process mining [1] algorithms use logs of traces to discover and analyse process models.

A business process is a dynamic system. Activities and their order are driven by unobservable inputs such as worker preferences, time, cost and competition pressures. Business processes are used to manage operations and ensure adherence to regulation, so changes may indicate developing problems or have legal implications and should be detected quickly. Conversely, if the change is valid, the process models need to be re-estimated to reflect the new reality.

To the best of our knowledge, this is the first study to consider process mining in non-stationary environments in an online manner, in a principled way. Business process research has discussed the need for flexibility and allowing for, and timely detection of, process change [2, 3, 4]. Questions of how much data is needed and how to identify when the process has changed, have been less investigated. In [5] statistical tests on features in log files are used to identify where in a log file the process changed. Here we build on our previous work [10] to propose a principled approach to efficiently mine and detect change to both model probabilities and structures, recovering the sequence of changed process models. The model changes we can detect are of more subtle nature than those detectable by re-estimation of standard process models, such as Petri nets.

2 Representations of Business Processes

In [8] we proposed a radically different view of processes as distributions over allowed strings of activities. Traditionally, business processes are often represented

*Phil Weber is supported by a Doctoral Training Grant funded by EPSRC and the School of Computer Science, University of Birmingham.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

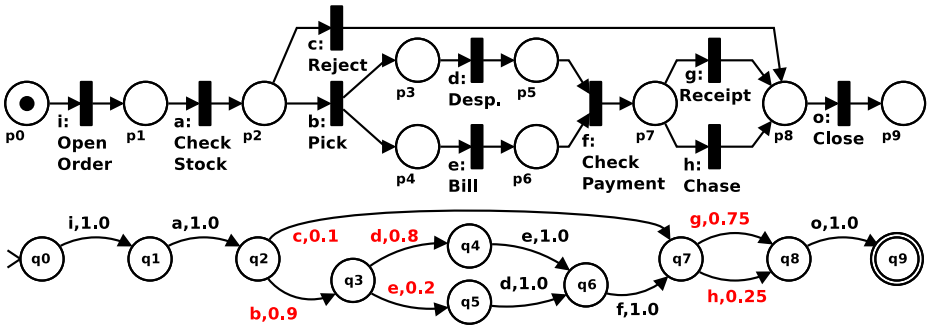

Fig. 1: Petri net showing a simplified order process, with equivalent PDFA labelled with symbols representing activities and their conditional probability.

by a restriction of Petri nets, Sound Workflow nets [7] (SWF-nets). An SWF-net $N = (S, T, W, M)$: T, S are finite sets of *transitions* and *places*, $T \cap S = \emptyset$, connected by arcs $W \subseteq (S \times T) \cup (T \times S)$. N has a single start and single end place, with every transition on a path between them. *Marking* $M : S \to \{0, 1\}$ describes the distribution of *tokens* over places and defines the state of the process. A transition may *fire* when there is a token in each of its input places, removing these tokens and adding a token to each of its output places. The initial (final) marking M_0 (M_F) is a single token in the start (end) place. When a process is started from marking M_0, all activities must be potentially executable, and the process must terminate in M_F. The *Reachability Graph* of N is the set of markings reachable from M_0 by firing a series of transitions.

SWF-nets concisely represent the process structure, but to detect change the important question is what traces can be seen, with what probabilities. We represent *activities* as symbols from a finite alphabet Σ, *traces* as strings $x \in \Sigma^+$. The true business *process* $\mathcal{M}$ is modelled by a probability distribution $P_{\mathcal{M}}$ over traces. Probability of trace x is $P_{\mathcal{M}}(x)$, such that $\sum_{x \in \Sigma^+} P_{\mathcal{M}}(x) = 1$. An 'event log' W is a finite multiset over Σ^+, drawn *i.i.d.* from $P_{\mathcal{M}}$. The task of a process mining algorithm is to learn from W a distribution $P_{\mathcal{M}'}$, to approximate $P_{\mathcal{M}}$.

We use probabilistic deterministic finite automata (PDFA) [6] to represent distributions generated by processes, as a common denominator to which other representations can be converted. PDFA A is a five-tuple $(Q_A, \Sigma, \delta_A, q_0, q_F)$. Q_A is a finite set of states; Σ an alphabet of symbols; $q_0, q_F \in Q_A$ single start and end states. $\delta_A : Q_A \times \Sigma \times Q_A \to [0, 1]$ defines the conditional transition probability function between states. $\delta_A(q_1, a, q_2)$ is the probability that given we are in state q_1, we parse a and arrive in state q_2: $\sum_{a \in \Sigma, q \in Q} \delta_A(q_1, a, q) = 1$.

PDFA A generates a probability distribution P_A on Σ^+. The probability of string x, $P_A(x)$, is found by multiplying the probabilities of the arcs followed to parse x on its unique path from the single start state q_0 to unique end state q_F. A PDFA can be obtained from the reachability graph of a SWF-net by labelling arcs with probabilities, e.g. from frequencies of activities in an log.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

3 Method for Model Estimation and Online Mining

In this first study we assume a real business process $\mathcal{M}$ generates traces into a log file W without noise, and that we can model this process as a PDFA. These assumptions allow process mining with the Alpha algorithm [9], which uses simple rules to construct an SWF-net from the data in W. The main idea is, since traces are generated randomly according to an unknown distribution, to use the behaviour of the mining algorithm and probabilities of traces to determine the minimum traces needed to be confident that the mining algorithm will create the correct model, and thus that if a different model is produced, the underlying model has changed, rather than being a feature of the sample.

To initially estimate $\mathcal{M}$, we use the most recent n_0 (a known over-estimate) traces from W. We convert the Petri net mined by Alpha to a PDFA by labelling its reachability graph with probability estimates derived from the frequencies of activities in the traces used for mining. The distribution that this automaton generates, $P_{\mathcal{M}}$, is the estimate of the underlying model $\mathcal{M}$.

Alpha uses relations between pairs of activities in the traces in the log file, to determine the structures to create in the Petri net. For two activities a, b from the set $\mathcal{A}$ of activities in the model; $a \to_n b$ (causal) if b always follows a in a log of n traces, never vice-versa; $a \#_n b$ (no relation) if a and b never follow each other; and $a \|_n b$ (parallel) if both ab and ba occur in the log. A sequence of two activities a and b is created when a is causally related to b and no other, i.e. $a \to_n b$ and $\forall c \in \mathcal{A}, a \not\to_n c \wedge c \not\to_n b$. An exclusive split from a to b_1, b_2 is when $a \to_n b_1$, $a \to_n b_2$, $b_1 \#_n b_2$; and a parallel split when $a \to_n b_1$, $a \to_n b_2$, $b_1 \|_n b_2$. Splits and joins with more activities are created similarly.

From these rules we derive formulae for the probability of Alpha discovering these relations from the log, and hence correctly mining process structures, based on the number of traces n, and the probabilities in the model. Let $\pi(ab)$ be the probability of ab occurring in a trace. We define $P_\alpha(a \to_n b)$ as 'the probability that Alpha infers relation $a \to_n b$ over n traces', similarly for the other relations. These formulae give the probabilities for Alpha discovering these relations[1] [8]:

$$\begin{aligned} P_\alpha(a \to_n b) &= \big(1 - \pi(ba)\big)^n - \big(1 - \pi(ab) - \pi(ba)\big)^n, \\ P_\alpha(a \#_n b) &= \big(1 - \pi(ab) - \pi(ba)\big)^n, \text{ and} \\ P_\alpha(a \|_n b) &= 1 - \big(1 - \pi(ab)\big)^n - \big(1 - \pi(ba)\big)^n + \big(1 - \pi(ab) - \pi(ba)\big)^n. \end{aligned}$$

The following inequalities assume that the discovery of the relations are independent of each other, and give close approximations to the probability of Alpha mining an XOR (1) and parallel (2) split correctly from a log of n traces [8]:

$$P_\alpha\big(a \to_n (b_1 \# \ldots \# b_m)\big) \leq \prod_{1 \leq i \leq m} P_\alpha(a \to_n b_i) \times \prod_{1 \leq i < j \leq m} P_\alpha(b_i \#_n b_j), \quad (1)$$

$$P_\alpha\big(a \to_n (b_1 \| \ldots \| b_m)\big) \leq \prod_{1 \leq i \leq m} P_\alpha(a \to_n b_i) \times \prod_{1 \leq i < j \leq m} P_\alpha(b_i \|_n b_j). \quad (2)$$

[1] We assume acyclic processes so ab and ba cannot occur together in a trace: $\pi(ab \wedge ba) = 0$.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

We combine these formulae for each structure by treating each structure as conditionally dependent on the structures before it in the model, to determine the probability of correctly mining the full model, e.g. for model PN in Fig.1,

$$P_\alpha(PN) = P_\alpha(i \rightarrow_n a) \times P_\alpha\big(a \rightarrow_n (b \#_n c) | i \rightarrow_n a\big) \times \\ P_\alpha\big(b \rightarrow_n (d \parallel_n e) | a \rightarrow_n (b \#_n c)\big) \times \ldots.$$

We use these formulae to obtain n such that when mining with n traces we will with probability $1-\epsilon$ return the Petri net corresponding to the underlying model M, for a desired confidence level $0 < \epsilon \ll 1$. Thus if mining produces a different model, we can have confidence $1-\epsilon$ that the underlying process has changed.

To detect change, we mine repeatedly from W_n, the most recent n traces from log file W, to obtain at each iteration a model $\mathcal{M}'$. To determine if the process has changed, we compare the distribution $P_{\mathcal{M}'}$ generated by $\mathcal{M}'$, with the ground truth $P_{\mathcal{M}}$. We use common statistical tests such as the Chi^2 test to detect if $P_{\mathcal{M}'}$ is significantly different from from $P_{\mathcal{M}}$.

Let T be the set of trace classes (unique traces) in W, and $m = |T|$. Since we consider only acyclic models, T is finite. For each $t_i \in T$, let $P_{\mathcal{M}}(t_i)$ be the probability of t_i under $P_{\mathcal{M}}$, and $n(t_i)$ the number of times t_i occurs in W_n. $n(t_i)$ is Binomially distributed and for large enough n, is approximately Normally distributed. The difference between the expected and sample counts for each trace class is also Normally distributed, so we test the sum of the differences:

$$\chi_s^2 = \sum_{i=1}^{m} \frac{\big(n(t_i) - nP_{\mathcal{M}}(t_i)\big)^2}{nP_{\mathcal{M}}(t_i)}, \text{ and } p = Pr(\chi_{m-1}^2 \geq \chi_s) = \int_{\chi_s^2}^{\infty} f(\chi_{m-1}^2) d(\chi_{m-1}^2),$$

where χ_s^2 is the sample Chi^2 statistic and $f(\chi_{m-1}^2)$ the density function of the Chi^2 distribution with $m-1$ degrees of freedom. The 'p-value' p gives the probability that the Chi^2 distribution would exceed the measured value, indicating that with probability $1-p$, the process has changed.

After detecting change, we wait for n traces, then re-estimate $\mathcal{M}$ and n.

4 Experimentation and Analysis

We used the PDFA in Fig.1 as the initial underlying model. 50 traces are needed for 99% confidence in mining the correct Petri Net, but as probabilities vary this can increase to 500. We tested a number of changes, results from a limited selection of which are shown in table 1: (a) probability of b, c set to $0.1, 0.9$, (b) prob. of d, e from q_3 set to 0.5, (c) parallel execution of d, e after b, changed to exclusive choice, (d) removed arc h from q_7. Columns 1-5) show that varying numbers of traces were estimated as necessary for mining. The changes were discovered in each case, with no false positives, i.e. incorrect detection of change. To correspond to the 99% confidence in mining, p-values below 0.01 were taken as significant. The Kullback-Leibler divergence between each changed and original estimate of the underlying model, is shown for comparison, but there was no clear correspondence between these values and change detection.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

	Optimal Sample				Large Sample			
Change	**Sample**	**Detect**	**KL**	**p-val**	**Sample**	**Detect**	**KL**	**p-val**
(a)	271	22	0.026	0.007	500	18	0.013	0.034
(b)	45	16	0.163	0.010	500	47	0.013	0.040
(c)	45	15	0.167	0.007	500	195	0.015	0.043
(d)	45	49	0.421	0.004	500	393	0.016	0.034

Table 1: Results for four types of change (section 4). 'Sample' traces were used for mining, change detected in 'Detect' iterations, 'KL' and 'p-val' record the Kullback-Leibler Divergence and Chi^2 p-value between new and previous estimate of underlying model. 'Optimal Sample' results used the method described for minimal sample size; 'Large Sample' used excessively large samples.

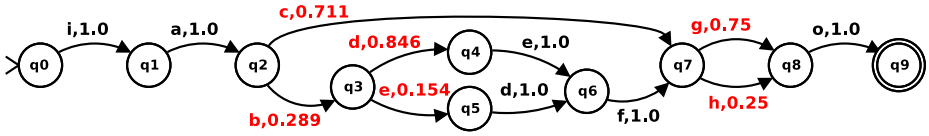

Fig. 2: A PDFA with same structure as Fig.1, but representing a significantly different probability distribution.

As a comparison, we repeated the experiment mining from logs of 500 traces. All changes were detected, but as columns 6-9 of table 1 show, in general many more traces elapsed before detection.

Since we wait before re-estimation so that the traces used for mining will all been drawn from the changed underlying model, a key contribution of our method is that the mined models show the sequence of changed process models over time. Also, in many cases change is significant, but only evident in the PDFA probabilities (e.g. Fig.2), whereas the Petri net structure is unchanged. So for change detection, a probabilistic modelling language such as PDFA seems more appropriate than a purely structural representation.

To simulate a fast-changing environment we tried re-estimating the model without waiting after detection. This results in false detections (Fig.3) until enough traces have been generated for the log to reflect the new model. The initially estimated model will be invalid as the log will contain a mix of traces from the old and new models. However, although we cannot say with confidence whether these changes are valid, we can suggest that there may have been a change, and that after the n traces estimated as needed for confidence $1-\epsilon$ in mining the correct model, have been generated, using p-value ϵ we can with confidence $1-\epsilon$ accept the next change detected as true.

5 Conclusion and Future Work

We presented a novel method for online mining of processes in non-stationary environments. Using a probabilistic view of processes and mining algorithms,

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

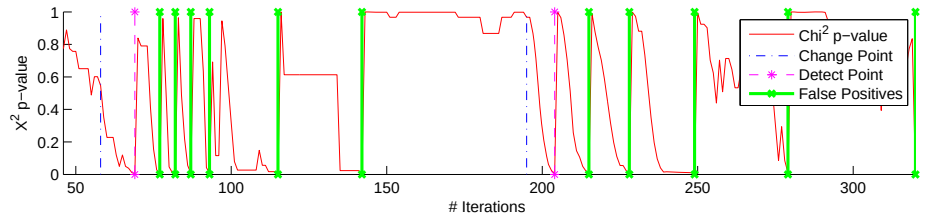

Fig. 3: Detection of true and false changes in fast-changing environment, with the model re-estimated immediately, rather than waiting, after change detection.

we can estimate (with a given confidence level) the number of traces needed for mining, enabling confidence that discovered change is true rather than an artefact of the log files. This allows us to recover the set of changed process models in use over time. Also, whereas process mining typically uses non-probabilistic representations such as Petri nets, we are able to discover change that is only apparent in the probabilities in the model, while the structure is unchanged. This first study leaves many open questions, such as whether the analysis can be applied to more refined process mining algorithms, noisy log files, or complex or unstructured processes. We hope to address some of these in future work.

References

[1] van der Aalst, W. M. P. and Weijters, A. J. M. M. Process mining: a research agenda. *Comput. Ind.*, 53(3):231–244, 2004.

[2] Rinderle, S., Reichert, M., and Dadam, P. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.

[3] Weber, B., Sadiq, S. W. and Reichert, M. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.

[4] Schonenberg, H., Weber, B., van Dongen, B. F. and van der Aalst, W. M. P. Supporting flexible processes through recommendations based on history. In Dumas, M., Reichert, M., and Ming-Chien Shan (eds.), *BPM 2008*. LNCS vol. 5240, pp. 51–66. Springer, 2008.

[5] Bose, R. P. J. C., van der Aalst, W. M. P., Zliobaite, I. and Pechenizkiy, M. Handling concept drift in process mining. In Mouratidis, H. and Rolland, C. (eds.), *CAiSE 2011*, LNCS vol. 6741, pp. 391–405. Springer, 2011.

[6] Vidal, E., Thollard, F., de la Higuera, F., Casacuberta, F., and Carrasco, R. C. Probabilistic Finite-State Machines - Part I. *IEEE Trans. Pattern Anal.*, 27(7):1013 – 25, 2005.

[7] van der Aalst, W. M. P. (1998). The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[8] Weber, P., Bordbar, B., and Tiňo, P.. A principled approach to the analysis of process mining algorithms. In Yin, H., Wang, W., and Rayward-Smith, V. J. (eds.), *IDEAL 2011*, LNCS vol. 6936, pp. 474–481. Springer, 2011.

[9] van der Aalst, W. M. P., Weijters, A. J. M. M., and Maruster, M. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[10] Weber, P., Bordbar, B., and Tiňo, P.. Real-Time Detection of Process Change using Process Mining. In Jones A. V., (ed.), *ICCSW 2011*, pp. 108–114, London, 2011.