Real-Time Detection of Process Change using Process Mining

Phil Weber, Behzad Bordbar, and Peter Tiňo

School of Computer Science, University of Birmingham, B15 2TT, UK.
{p.weber,b.bordbar,p.tino}@cs.bham.ac.uk

Abstract. Process Mining is the discovery of business processes from log files. One application is ensuring conformance to prescribed processes or business rules. Businesses operate in real time, needing to quickly react to change. We consider requirements for process mining to support this: a notion of real time, and methods to compare processes and detect significant change. We present initial results confirming the validity of the approach.

Keywords: Process mining, machine learning, real time, distributions.

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 probabilistic automaton. Each directed arc is labelled with a symbol representing an activity, and the conditional probability of that activity taking place next.

As the process is executed, the systems involved will record information in log files. Abstracting from detail, the 'trace' of a single enactment of this process might be recorded as a string of symbols $iabdefgo$. Process mining [6] algorithms use logs of such traces to discover and analyse process models.

Business processes are used to manage business operations, which today take place in real time, under pressures of time, cost and competition. Processes may also ensure adherence to business rules or regulatory requirements. Divergence from these processes may therefore indicate a business problem, or have legal ramifications, and so such changes need to be detected in a timely manner.

To enable detection of process change in real-time, several requirements need to be addressed. Firstly, a definition of real time and its application to process mining; secondly, a method to measure accurately the difference between two processes; thirdly, a method to detect change in a process; and finally a notion of statistical significance of the change. We briefly address these points in the following sections, after first introducing a probabilistic view of business processes and process mining, which underpins the subsequent ideas.

2 A Probabilistic View of Business Processes

We model *activities* as symbols from a finite alphabet Σ, *traces* as strings $x \in \Sigma^+$, and a *process* as a probability distribution $P_{\mathcal{M}}$ over traces. Probability of

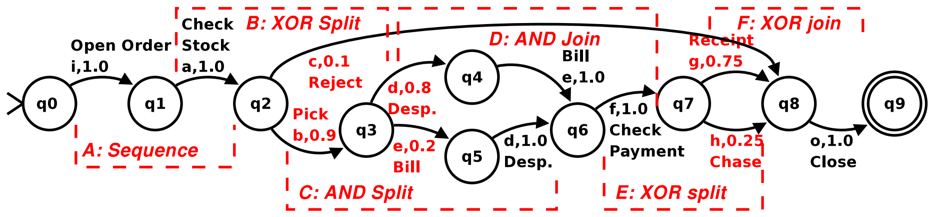

Fig. 1. PDFA showing a simplified business process for fulfilling an order.

trace x is $P_{\mathcal{M}}(x) : \sum_{x \in \Sigma^+} P_{\mathcal{M}}(x) = 1$. The task of a process mining algorithm is to learn a distribution $P_{\mathcal{M}'}$, to approximate $P_{\mathcal{M}}$, from the finite log W drawn *i.i.d.* from $P_{\mathcal{M}}$. This differs from existing views of process mining, which focus on discovery of a model structure in a specific representation such as Petri nets.

We use probabilistic deterministic finite automata (PDFA) [8] (Fig.1) to represent the probability distributions generated by process models, as a common denominator to which processes in other representations can be converted. A PDFA is a five-tuple $A = (Q_A, \Sigma, \delta_A, q_0, q_F)$, where Q_A is a finite set of states; Σ an alphabet of symbols; $q_0, q_F \in Q_A$ the single start and end states; and $\delta_A : Q_A \times \Sigma \times Q_A \to [0, 1]$ is a mapping defining the conditional transition probability function between states. $\delta(q_1, a, q_2)$ is the probability that given we are in state q_1, we parse a and arrive in state q_2. Given a current state and symbol, the next state is certain. The probabilities on arcs from a state sum to 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.

3 Overview of Approach

3.1 Real-time Process Mining

The term 'real time' is used subjectively of systems which appear to process information 'fast'. Formally, real-time systems 'must react within precise time constraints to events in the environment' [2]. The key is predictability and results guaranteed in a specified time, rather than speed. For us this means identifying process change as soon as possible, but with confidence that change is significant.

We consider two main constraints: accuracy and time. The mining algorithm must produce a model 'close' to the 'true' model using some notion of distance between distributions. We expect accuracy to increase with the amount of data, but for this to increase mining time. So these two constraints act in tension. We desire to minimise mining time, but characteristics of the ground truth distribution will determine the minimum data needed for confidence in mining accuracy.

This lower bound ensures we use the correct baseline, against which to measure change. We do not consider issues such as predicting time to detect change, from the type or magnitude of change. Other environmental issues may also affect the real time behaviour of the system [2] and need to be taken into account.

3.2 Determining the Amount of Data Needed for Mining

One way to determine the amount of data needed is to consider the structures in a process (highlighted in Fig.1), and the probability of an algorithm discovering these structures. In [9] we discuss this approach and apply it to the Alpha algorithm [7], which uses heuristics about the relations seen between pairs of tasks in the log, to construct a Petri net. To compare this non-probabilistic model against the ground truth distribution, we convert the Petri net to a PDFA by labelling its reachability graph (state model) with maximum likelihood probabilities obtained from the mining log. This allows us to satisfy the accuracy constraint. We do not address the time constraint as Alpha is relatively simple.

3.3 Methods to Detect Process Change

We mine repeatedly from sublogs, using a 'sliding window', and compare the distribution generated by the mined model with the ground truth distribution. There are many measures of difference between probability distributions, such as Euclidean distance, Kullback-Leibler Divergence. Some can be efficiently calculated from PDFA, but it is not clear what distance is statistically significant. Instead, we use statistical tests for detecting that the mined distribution, or its PDFA representation, has changed significantly from the ground truth.

The count of each unique trace x in the log can be modelled as a Binomially-distributed random variable, since any trace in the log will either be x, or not. The same is true of the number of times each arc in the PDFA is used in generating the log: each trace will either use that arc, or not (at present we assume acyclic models). If the number of traces is large enough relative to the trace/arc probabilities, the Binomial can be approximated by the Normal distribution.

Goodness of Fit Test on the Distribution: The sum of k Normally distributed random variables follows a Chi^2 distribution with $k-1$ degrees of freedom. Thus we can use the Chi^2 test to determine whether the difference between the count of each unique trace found in the sample, and the expected count, is likely under the assumption that the log was drawn from the ground truth distribution. The so-called p-value gives the probability that the Chi^2 distribution would exceed the measured value, indicating that with probability $1-p$, the process has changed.

Bounds and Hypothesis Tests on the PDFA: We expect the PDFA from the mining result to have the same state structure as the ground truth PDFA (making assumptions about the ground truth PDFA and the mining algorithm). The Hoeffding inequality upper bounds the probability of a sum of random variables deviating from its expected value. As [3], we use this to compare the probability of each arc from equivalent states in the models, by comparing the sum of the Bernouilli variables that each trace involves use of that arc.

Secondly, as [4] we use a hypothesis test to test how likely it is that an arc would be used the number of times indicated by its probability in the mined model, to generate the log, assuming the ground truth probability. Here the count is modelled as Binomial or Normal variable.

Bounds and Hypothesis Tests on Traces: The methods described for testing PDFA arcs can similarly be applied to process traces, so that we can use Hoeffding bounds or hypothesis tests to determine whether a process trace is likely to occur with the observed frequency, under the ground truth distribution.

4 Experimentation and Analysis

We used the example process of Fig.1. Using our method [9] the Alpha algorithm needs 44 traces to, with 99% probability, correctly mine a (non-probabilistic) Petri net with the correct structure. We randomly simulated the PDFA to produce an MXML[1] format log file of this size, and regularly updated it by simulating one new trace and removing the oldest. This simulates a 'sliding window' onto a log file being updated in real time by a live process. Changes were introduced to the probabilities or structures in this PDFA. At each iteration, we used the Alpha algorithm[2] to mine a Petri Net from the current log and converted to a PDFA (section 3.2). We recorded distances between the distribution generated by this PDFA and the ground truth, and results of the tests in section 3.3.

We ran three experiments to test the hypotheses that (i) change is detectable using a variety of methods, (ii) more significant change is detected in fewer traces, and (iii) the predicted number of traces for mining the model is the optimum to use for detecting change, thus allowing detection in real time.

Since the Alpha algorithm mines only a Petri net structure (no probabilities), it needs a relatively small sample of traces, which exhibits high variance from the ground truth (Fig.2), resulting in high risk of false positives (incorrectly detecting change) or false negatives (not detecting true change). We did not take this into account beyond ensuring no false positives occurred before change was introduced, but it would affect the detection point. These initial results were also based on one test only of each sample. The main results seem clear, but are not statistically valid without averaging over multiple tests.

Varying Probabilities We varied probabilities in the XOR split B, and parallel split C. Small variations (< 0.1) were not detectable, although the distance measures increased. For the XOR split, change to $p(ab) = 0.7$ was discovered in 28 iterations, reducing to 9 for $p(ab) = 0.1$. Detection was first by the hypothesis test on strings (Fig.4) or arcs, then by X^2 (Fig.3), and last by the Hoeffding tests. The looseness of the Hoeffding bound allows the string/arc frequencies to be more readily accepted as within confidence bounds given the ground truth.

The variation of AND probabilities was tested with probability of the structure in the model being 0.9 and then 0.1. The latter change was detected first by arc differences (Fig.5), the string difference methods not detecting it at all. This is explained by the probability of traces passing through the AND structure being too low to detect significant changes, but for those that do, changes to arc usage are local and not affected by the global probability of the structure.

[1] Mining eXtensible Markup Language, see `www.processmining.org`.
[2] implemented in the process mining tool ProM (`www.processmining.org`).

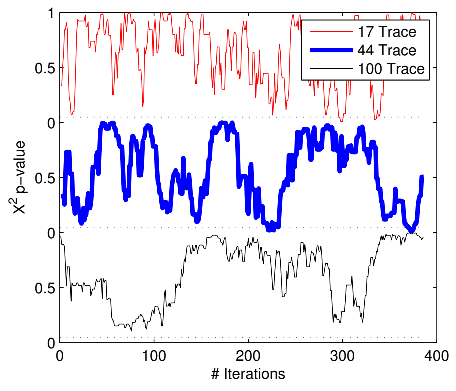

Fig. 2. Fluctuations in X^2 p-value over time, from unchanged source process.

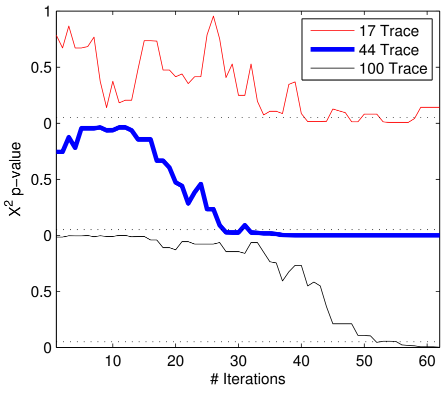

Fig. 3. Detection of XOR probability change using X^2 p-value.

Varying amount of data We varied the amount of data in the 'sliding window'. With 44 traces we see high variance in the probability distribution, seen in the large fluctuations in X^2 p-value in the centre graph of Fig.2. The lower graph shows that the frequency and amplitude of these changes is reduced with 100 traces, with no significant (0.05) p-values. The cost is slower detection of change (Fig.3 and 4). Conversely, reducing the number of traces to 17, change can be detected sooner, but with higher risk of false positive or false negative.

5 Related Work

Process Mining has been an active area of research since the early 1990s, recently reviewed in [6]. Non-probabilistic representations such as Workflow nets are commonly used [7]. Probabilistic approaches are the exception. 'Real-time' is used informally in Business Process research in regard to the need for flexibility and process change to respond to a changing environment [5].

An examination of the literature pertaining to real-time data mining, and stream mining, may inform improvements to our method. Concept drift is the detecting change of change in machine learning. In [1] this is discussed in a process mining context, focussing on model structure rather than probability.

6 Conclusion and Future Work

We examined various methods for detecting change in a running process, with initial results showing that using the optimal amount of data to be confident that the mined process is correct, various statistical methods can be used to efficiently detect change in real time.

More work is needed to address the effect of variation in the underlying distribution and the risk of falsely identifying or missing change, and to predict

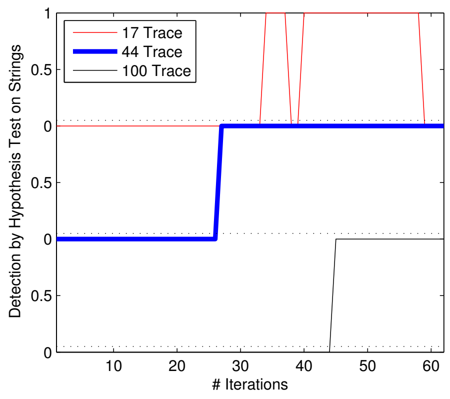

Fig. 4. Detection of XOR probability change using hypothesis test on strings.

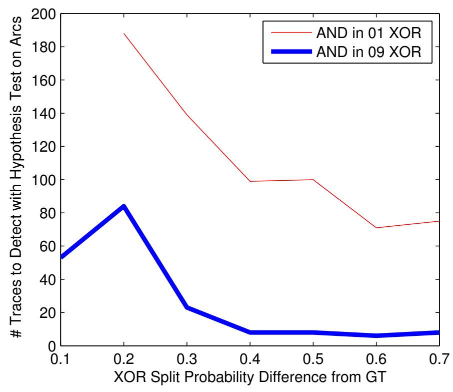

Fig. 5. AND change detection using hypothesis test on arcs, varying probabilities.

the time to detect change. Some distances between distributions can be efficiently calculated from PDFA, so understanding of the significance of distance measures, would lead to more efficient methods for detecting change.

References

1. Jagadeesh Chandra Bose, R. P., 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*, LNCS, vol. 6741, pp. 391–405. Springer, 2011.
2. Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, Norwell, USA, 1997.
3. Carrasco, R. C. and Oncina, J. Learning stochastic regular grammars by means of a state merging method. In Rafael C. Carrasco and José Oncina (eds.), *ICGI*, LNCS, vol. 862, pp. 139–152. Springer, 1994.
4. Jacquemont, J., Jacquenet, J., and Sebban, M.. Mining probabilistic automata: A statistical view of sequential pattern mining. *Machine Learning*, 75(1):91–127, 2009.
5. Rinderle, S., Reichert, R., and Dadam, P.. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.
6. Tiwari, A., Turner, C. J., and Majeed, B. A Review of Business Process Mining: State-of-the-Art and Future Trends. *Bus. Process Manage. J.*, 14(1):5 – 22, 2008.
7. van der Aalst, W. M. P., Weijters, T., and Maruster, L. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–42, 2004.
8. 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.
9. P. Weber, B. Bordbar, and P. Tiňo. A principled approach to the analysis of process mining algorithms. In *Proceedings of the 12th International Conference on Intelligent Data Engineering and Automated Learning (to appear)*, 2011.