Bayesian classifcation of events for task labeling using workfow models

Buffett, Scott; Geng, Liqiang

Publisher's version / Version de l'éditeur:
https://doi.org/10.1007/978-3-642-00328-8_10
Business Process Management Workshops, pp. 97-108, 2008-09

*Bayesian Classification of Events for Task Labeling Using Workflow Models**

Buffett, S., Geng, L.
September 2008

* published at The 4th Workshop on Business Process Intelligence (BPI 08) to be held in conjunction with Business Process Management (BPM 2008). Milan, Italy. September 1, 2008. NRC 50389.

Bayesian Classification of Events for Task Labeling Using Workflow Models

Scott Buffett and Liqiang Geng

National Research Council Canada, Fredericton, NB, E3B 9W4
{Scott.Buffett,Liqiang.Geng}@nrc.gc.ca

Abstract. We investigate a method designed to improve accuracy of workflow mining in the case that the identification of task labels for log events are uncertain. Here we consider how the accuracy of an independent task identifier, such as a classification or clustering engine, can be improved by examining workflow. After briefly introducing the notion of iterative workflow mining, where the mined workflow is used to help improve the true task labelings which, when re-mined, will produce a more accurate workflow model, we demonstrate a Bayesian updating approach to determining posterior probabilities for each label for a given event, by considering the probabilities from the previous step as well as information as to the beliefs of the labels that can be gained by examining the workflow model. Experiments show that labeling accuracy can be increased significantly, resulting in more accurate workflow models.

Key words: workflow, process mining, task labeling, Bayesian classification

1 Introduction

In recent years, research in business process management has seen a considerable effort in the field of workflow mining. Workflow mining involves automatically (or semi-automatically) inspect a log of machine-level events executed by a number of people, working together on one or more business processes within an enterprise, and to discover and identify workflows inherent in the activity. Such discovered workflows can then be analyzed to determine such aspects as common sequences of events or common communications between parties within the workflow of the process. Such analyses can then be considered in order to determine how the process can be made more efficient, or instead be compared with ongoing activity to ensure employees are complying with the common workflow. To accomplish this effectively, there are number of interesting problems that have been investigated in the recent literature. These include algorithms for discovering causal relations in activates and complex constructs [4, 3], efficient methods for analyzing large logs [3], and user friendly visualization of discovered workflow [12], conformance analysis between discovered models and the traces [6, 10], among others.

One thing that much of the literature has failed to address is, however, the difficulty in simply identifying machine-level events as the high-level tasks they represent. That is, most research assumes that an accurate labeling of tasks is already obtained or is easily determined. However, in many practical situations, especially where there is not a workflow management and transaction system available, this is not actually the case. For example, how can a system recognize that an e-mail message confirming a travel itinerary for a business trip to Paris for one employee is the same task (although in a different instance) as a message confirming travel plans for a trip to Tokyo for another? Each event represents the same task conceptually in the grand process of travel planning, however the similarity would not be easily recognizable to a machine, due to the fact that the two messages contain a lot of differing information. Recent work has focused on solving this problem by inspecting keywords common to messages, and using machine learning classification to decide whether to label the two events as the same task [8]. This could be effective in this situation, since both messages might include words such as “travel”, “itinerary” and “confirm”. However, there will likely be a number of errors with these methods, since for example the traveler could send a message asking an administrative staff member to “please confirm that you received my requested travel itinerary”. This could also be mistakenly labeled as an itinerary confirmation.

The goal of this paper is to present a method for assisting such a classification method in an effort to reduce the error rate. To do this, we consider the current workflow mined, and use this to add new information to the classification engine in an effort to increase its accuracy. Using the updated task labeling, one could have the workflow mined again if it is believed to be inaccurate due to mislabeling of tasks, yielding a process where the workflow essentially refines itself. Or rather than refining the workflow, this technique could also be useful in the case where the workflow model is already sufficiently accurate, and the task is simply to monitor new activity to ensure that it is compliant with the accepted model. Improving the task labeling accuracy here should reduce the number of false positives and false negatives in the compliance checking phase. Consider again the travel confirmation scenario, where the event was erroneously classified as an itinerary confirmation message. One might be able to look at the frequencies in the model and determine that, in a high percentage of cases, such an event will follow the actual booking of the travel[1]. In this case, however, it can be determined easily that no such activity took place previously (since the employee was just checking to see if her request was received), thus providing strong evidence that this event should be classified as something else. To enable the classification engine to consider this new evidence and make more informed decisions when determining how to label an event, we employ the concept of *Bayesian updates*. Bayesian updates are useful for refining the probabilities of events when both prior likelihoods and new probabilistic evidence are available.

The paper is organized as follows. In section 2 we discuss some required concepts, including workflow mining and Bayesian classification. In section 3 we

[1] One might think that this would be the case 100% of the time, but it might be that in a small number of cases the employee may have booked travel herself over the phone, and thus there would be no evidence of this activity in the log.

introduce the notion of iterative workflow mining, and discuss the concept of the belief state in workflow modeling. In section 4 we demonstrate how Bayesian classification can be used to update probabilities derived by the classification engine, given probabilities determined from given constructs found in the workflow model. Section 5 then demonstrates the effectiveness of our approach with some empirical results. Section 6 then discusses conclusions and related work, while section 7 offers a few directions for future work.

2 Background

2.1 Workflow Mining

Workflow mining [3] refers to the process of autonomously examining a transaction log of system events, and extracting a model of the underlying process being executed by the events. Generally speaking, an log consists of a number of events, each of which being associated with a *task* and a *case*. An event's task refers to the actual activity the event represents, while the event's case refers to the instance of the underlying business process to which the event belongs. Each case in the log consists of a sequence of tasks (often referred to as a *trace*) that represent a complete and ordered set of actions that are executed in an instance of the business process. Workflow mining techniques are then used to build a model of the business process by representing the different ways a case in the process can be executed. A number of different representations have been used in the literature, A number of different representations have been used in the literature, such as directed acyclic graphs [4], finite state machines [6], variations of Bayesian networks [11], workflow schemas [7], and Petri Nets [9, 2]. Since Petri Nets can easily represent the most common constructs of workflows, such as sequences, parallelism, iterations, and mutual exclusiveness, in this paper, we adopt the Petri net as our representation of workflow models. However, it should be noted that our ideas on iterative refinement of models are independent of the model representations, and thus the method can be applied for different models. Figure 1 represents a small example log, as well as the resulting Petri net representing the mined workflow. Any legal sequence of transitions that takes a token from the start (leftmost) place to the end (rightmost) place represents a different way of executing the business process.

(*Case, Task*)		
1. (1, A)	9. (4, B)	17. (3, E)
2. (1, B)	0. (3, D)	18. (6, A)
3. (2, A)	11. (2, B)	19. (5, B)
4. (2, C)	12. (2, E)	20. (4, E)
5. (1, C)	13. (3, B)	21. (6, D)
6. (3, A)	14. (4, D)	22. (6, B)
7. (1, E)	15. (5, A)	23. (5, E)
8. (4, A)	16. (5, C)	24. (6, E)

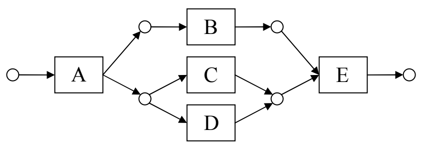

Fig. 1. Example log and corresponding workflow diagram.

2.2 Bayesian Classification

Bayesian classification is a technique from the field of supervised machine learning where objects are assigned to classes based on the likelihoods of the observed attributes or *evidence*. Given a set of classes, a Bayesian classifier is provided with information on the attributes of objects that belong to each class. When presented with new unclassified objects, the classifier makes a decision on which class is most likely to include the object. Consider a Bayesian classifier that classifies documents into one of two classes: literary and scientific. The classifier uses information on the likely attributes of members of each class (e.g. a document containing the word "hypothesis" is more likely to be from the scientific class). This probabilistic information can be obtained by observing the classification of several objects where the classes are known, and noting the frequency at which objects with certain characteristics are assigned to each class.

The probability model for a Bayesian classifier is as follows. Let $C*$ be the set of classes. The probability of an object belonging to a class $C \in C*$ given the observed evidence E is denoted by $P(C|E)$. This can be computed using

$$P(C|E) = \frac{P(C) \times P(E|C)}{P(E)} \tag{1}$$

where $P(C)$ is the prior probability an object belonging to class C, $P(E|C)$ is the probability of observing E given that the object belongs to C, and $P(E)$ is the probability of observing E. Returning to the document classification example, consider the initial observations that 60% of the documents are scientific (S) as opposed to 40% literary (L), and 70% of all scientific documents contained the word "hypothesis" as opposed to 5% in literary. Then the probability $P(E)$ of observing the evidence word "hypothesis" is $P(E|S)P(S) + P(E|L)P(L) = (0.7)(0.6) + (0.05)(0.4) = 0.44$, and thus the probability of a document containing "hypothesis" belonging to the scientific class is $\frac{0.6 \times 0.7}{0.44} = 0.95$, where the probability of such a document belonging to the literary class is $\frac{0.4 \times 0.05}{0.44} = 0.05$.

3 Belief States in Workflow Models

3.1 Iterative Workflow Mining

While it is not the central focus of this paper, in order to put the idea of Bayesian classification of task labels into context, we briefly introduce the concept of iterative workflow mining.

Iterative workflow mining refers to the process of modeling workflow by beginning with an initial model, and refining that model over a number steps. The technique is particularly useful when task labelings are uncertain (as is the case in this paper) and thus the best labeling and corresponding workflow model must be determined simultaneously. However, this process can also be effective in situations where labels are present but particularly difficult workflow constructs are known to exist in the conceptual model, and it is desired to manually model these initially and then refine the model based on the log traces. The process works as follows. Given the initial log, labels are obtained for a subset $C' \subseteq C$ of cases in the log (where possibly $C' = C$). We can think of this initial labeling as sort of training set, perhaps provided manually by a participant familiar with the business process being modeled. At any step in the process, there is a set $\mathcal{W}$ of possible workflow models, with a probability distribution $p : \mathcal{W} \rightarrow \Re$ indicating the likelihood of each being the actual model for C. So in the initial iteration, the set of workflows could be one or more that are initially manually specified, or in the case that C' is a proper subset of C, could consist simply of the workflow mined for C'. In each iteration, the set $\mathcal{W}$ is refined by considering the log using various means, as well as the set of workflows from the previous step. The process can be halted at any time, but will typically continue until one workflow emerges or the refining process used is considered complete, with the candidate in $\mathcal{W}$ with the highest probability being chosen as the final workflow.

3.2 The Refining Process

In order to give necessary background for the concept of belief states in workflow models, we give a short description of the refining process used in iterative workflow mining with incomplete task labeling. Let $\mathcal{W}$ and $p_{\mathcal{W}}$ be the set of workflow models and probabilities in a given iteration, let C be a set of cases in the log, and let L_c be the set of possible labelings for case $c \in C$. Each $\ell_c \in L_c$ maps each event in c to a member of the set T of possible tasks, and has probability $p_{L_c}(\ell_c)$ of being the correct labeling. Finally, let $p_x : T \rightarrow \Re$ be an initial probability distribution indicating the probability that an event x in the log actually represents a task $t \in T$. This could be provided by some independent process such as a machine learning classification engine. Initially this will induce the set of L_c over C. The different possible labelings in L_c for each case then induces a number of possible sets of case labelings, each with probability computed using $p_{L_c}(\ell_c)$ for each L_c. Denote this set of sets of cases as $\mathcal{C}$. The set $\mathcal{W}$ of possible workflows is then those workflows mined from each set of traces in $\mathcal{C}$, and the probability $p_{\mathcal{W}}(W)$ for $W \in \mathcal{W}$ is equal to the probability of the corresponding $C \in \mathcal{C}$. The idea is to iteratively refine the likely labelings for each event in the log using the workflow models, and to refine the workflow models using the likely event labelings. This should converge to a set of cases and corresponding workflow model that fit best.

3.3 Belief States

Let $\mathcal{W}$ be a set of workflow candidates. The belief state $B(\mathcal{W}, x)$ for a given log event x gives an indication of the possible locations in each workflow that the event x is likely to reside. In other words, it answers the question "Given that x is being executed, what is the current step in the business process being modeled?" More formally, $B(\mathcal{W}, x)$ gives a probability distribution function over the set of all transitions in all possible workflows, indicating the likelihood that

x represents that transition in that corresponding workflow. This, along with the probability distribution $p_{\mathcal{W}}$ over $\mathcal{W}$ can then be used to determine the probability $P(x)$ that x represents a particular task.

Consider an example where there is one possible workflow, which is depicted in Figure 2. Then a possible belief state for an event x that is believed to follow either task B or C and precede task I in its trace is specified by the probabilities that label the transition nodes. It is the goal of this paper to show how to use the belief state to update the label probabilities for the event being observed, which is in turn then used to update the belief state.

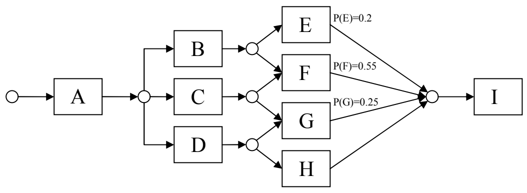

Fig. 2. Example Petri net showing the belief state for an event x that is believed to follow either task B or C and precede task I.

4 Bayesian Updates in Classification

4.1 Problem Setup

Let x be an observed event, and let $B(\mathcal{W}, x)$ be the belief state for x, indicating the likelihood of each task labeling for x. Next, let E denote the evidence obtained from the classification engine. E indicates which task t should label the observation (perhaps with some probability), where this classification is determined using methods independent of the workflow model or frequency tables, such as by inspecting the key words in an email, for example. The goal is to update the prior probabilities and determine a new probability $P(t|E)$ which takes into account the evidence generated by the classification engine.

In this section we demonstrate how Bayesian updating can be used to refine the probability distributions in two different scenarios: (1) when the evidence simply consists of a suggested task classification, and (2) when the evidence suggests a probability distribution over the set of tasks.

4.2 Task Labeling as Evidence

Let x be an event with belief state $B(\mathcal{W}, x)$, and let t_ℓ be the task selected for x by the classification engine. We observe the following:

1. For each $t \in T$ the probability $P(E|t)$ of observing the evidence E generated by the classification or estimation engine, given that the event is actually t. In other words, this is the likelihood that t would be labeled as t_ℓ, which gives

an idea of the performance of the engine. For clarity, this will be denoted as $P(t_\ell|t)$. This data may have to be obtained from historical data that has been verified, much like a training set, or could instead be provided by the classification system.

2. The probability $P(E)$ of observing E, given the belief state for the event. In other words, this is the likelihood that the event would be labeled as t_ℓ, and will be denoted as $P(t_\ell)$. This value can be computed by observing the probabilistic log labelings.
3. For each $t^j \in T$ the probability $P(t^j)$ that the observed event would be t, as dictated by the belief state.

Once these statistics are obtained, the probability that the observed event is truly task t, given that it is labeled as t_ℓ, is computed using Bayes' Rule:

$$P(t|t_\ell) = \frac{P(t) \times P(t_\ell|t)}{P(t_\ell)} \tag{2}$$

We demonstrate this with a brief example. Consider again the example workflow represented by the Petri net in Figure 2 with belief state specified for E, F or G. Suppose the classification engine indicates that event x should be labeled as task E, and the performance data compiled by the classifier specifies that the probability of returning E when each of the three tasks is truly the case to be 0.6, 0.3 and 0.1, respectively. Finally, it is determined that probability that a task here would be labeled as E is 0.31. This data, as well as the posterior probabilities for the true outcome of the task, are outlined in Table 1. The rightmost column in the table shows that there is actually only a 0.387 probability that the event is task E, compared to 0.532 for task F. This indicates that the classifier has not performed strongly enough in the past when classifying events as E's and thus, in this situation, we would be more likely to accurately identify the event if it was labeled as F.

t	$P(t)$	$P(E_\ell|t)$	$P(E_\ell)$	$P(t|E_\ell)$
E	0.20	0.60	0.31	0.387
F	0.55	0.30	0.31	0.532
G	0.25	0.10	0.31	0.081

Table 1. Computation of posterior probabilities as in the above example

4.3 Probability Distribution over Task Labelings as Evidence

The problem becomes more interesting, yet more complex, when the classifier returns a probability distribution over the set of possible tasks. In this case, rather than labeling the event as a B, the classifier could instead indicate that its belief that each task should label the event with some probability. This is different from the previous section since, rather than having the classifier suggest a label and compiling statistics regarding the accuracy of the label (e.g. that 45% of A's are labeled as A, etc.), the classifier suggests a distribution over the labels (e.g. that there's a 65% chance the event is an A, etc.). There are now two independent probabilities associated with each label, the priors and the classifier probabilities. The focus here is to present a method for using the classifier probabilities to update the prior probabilities.

Let $P_E(t)$ be the probability for task t given the evidence E found by the classification engine. In other words, the classifier found evidence E and determined that the probability of the event being t is $P_E(t)$. Also, let $P_c(t)$ be the likelihood of t as assumed by the classifier before any evidence is considered. If the classifier assumes initially that all tasks are equally likely, then simply $P_c(t) = 1/|T|$. The goal is to update the prior probability $P(t)$ from the belief state using $P_E(t)$, giving the posterior probability $P(t|E)$. Again, using Bayes' Rule, the posterior probability of t truly being the correct task labeling, given the evidence E obtained by the classification engine is

$$P(t|E) = \frac{P(t) \times P(E|t)}{P(E)} \tag{3}$$

The difficulty here lies in that $P(E|t)$ and $P(E)$ are not readily available, but rather must be computed. To do this, we show how this $P(E)$ can be expressed in terms of $P(E|t)$, eliminating that variable and producing a simplified expression that can be computed in terms of the prior probabilities and the classifier probabilities. We begin by expanding $P(E)$ to the following:

$$P(E) = P(E|t)P(t) + P(E|\neg t)(1 - P(t)) \tag{4}$$

where $P(E|\neg t)$ is the probability of obtaining E in the case that the event is anything but t. It is not possible to determine $P(E|t)$ and $P(E|\neg t)$ given the data available. However, one can determine the ratio between the two, and express one in terms of the other. Let x_t be this ratio. In particular, let

$$P(E|\neg t) = x_t P(E|t) \tag{5}$$

We derive x_t as follows. From the classifier probabilities, we know that $P(E|t)$ percent of events that are truly task t make up $P_E(t)$ percent of events that yield E. Likewise, $P(E|\neg t)$ percent of events that are truly not task t make up the other $1 - P_E(t)$ percent of events that yield E. Let n_t be the number of tasks that are truly t as assumed by the classifier before evidence is observed, let $n_{\neg t}$ be the number of tasks that are truly not t as assumed by the classifier before evidence is observed, and let n_E be the number of events that yield E. Then

$$\frac{P(E|t)n_t}{P(E|\neg t)n_{\neg t}} = \frac{P_E(t)n_E}{1 - P_E(t)n_E} \tag{6}$$

Since $n_t/n_{\neg t} = P_c(t)/(1 - P_c(t)$,

$$\frac{P(E|t)P_c(t)}{P(E|\neg t)(1 - P_c(t))} = \frac{P_E(t)}{1 - P_E(t)} \tag{7}$$

Solving for $P(E|\neg t)$, we get

$$P(E|\neg t) = \frac{(1 - P_E(t))P_c(t)}{P_E(t)(1 - P_c(t))} P(E|t) \tag{8}$$

Combining with equation 5 gives

$$x_t = \frac{(1 - P_E(t))P_c(t)}{P_E(t)(1 - P_c(t))} \tag{9}$$

which can be computed using the data given. Returning to equation 4, we can combine with equation 5 to obtain an expression in terms of $P(E|t)$:

$$P(E) = P(E|t)P(t) + x_t P(E|t)(1 - P(t)) \tag{10}$$

Replacing $P(E)$ in equation 3 then finally gives an expression that can be computed using the given prior probability of t and x_t:

$$P(t|E) = \frac{P(t)P(E|t)}{P(E|t)P(t) + x_t P(E|t)(1 - P(t))} = \frac{P(t)}{P(t) + x_t(1 - P(t))} \tag{11}$$

We demonstrate this approach with a brief example. Consider computing the posterior probability that the task is B. Suppose the prior probabilities and classification probabilities are such that $P(B) = 0.2$ and $P_E(B) = 0.7$. Also assume that the classifier assumed that all tasks initially equally likely, and there are 3 such tasks and thus $P_c(B) = 1/3$. First, x_B is computed:

$$x_B = \frac{(1 - P_E(t))P_c(t)}{P_E(t)(1 - P_c(t))} = \frac{(0.3)(1/3)}{(0.7)(2/3)} = 0.21$$

Next, the value for x_B is plugged in the simplified Bayes' equation, giving

$$P(B|E) = \frac{0.2}{0.2 + (0.21)(0.8)} = 0.54$$

So, considering the data and the two independently computed probabilities that the event is a task B, 0.2 (the prior probability based on workflow data) and 0.7 (the new evidence based on classification data), we determine that the true posterior probability that the event is indeed a B is 0.54.

5 Results

We demonstrate the effectiveness of the label probability updates by presenting a few results for a simple example. In particular, we validate the performance of the procedure by showing how labeling accuracy of tasks is improved. Improved labeling accuracy should then, in turn, result in more accurate workflow models. For simplicity, we consider the case where the classification engine provides suggested labelings for each event, as described in section 4.2. Taking into account the probability of each possible labeling, will only improve accuracy, so validation of the former case should be sufficient to confirm the procedure's effectiveness.

We used a simulated log file containing 1,000 cases, where each case was one of $ABDG$, $ABEG$, $ACEG$ or $ACFG$, yielding the workflow model demonstrated by the Petri net in Figure 3. This is then the unknown model of the underlying process that is to be discovered. Also a simulated classification engine was used to provide the initial labeling of the tasks. This simulation worked by taking the true task label for an event as input, and giving a random label as output based on some associated distribution, allowing us to simulate errors in the classifier. We programmed the classifier to label tasks A, B, C and G with 100% accuracy, and produce errors when attempting to label events that represented tasks $D-F$ (often mistakenly labeling a D as an E or an F, E as D or F, and F as D or E.). Thus we narrow the focus on simply attempting to improve the accuracy of the $D-F$ event labelings.

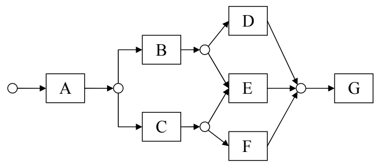

Fig. 3. Workflow model used in experiments.

Figure 4(a) compares the accuracy of the initial labeling with the updated labeling over the 1000 log cases, given different levels of accuracy for the classifier. This shows that the updated labeling performs extremely well, at all levels of task labeling accuracy. The performance is particularly good when initial labeling is especially poor, with double the accuracy when initial labeling accuracy approaches 0.33, which represents random labeling performance. This comparison is a bit unfair, since it can be easily computed that F is not in the belief state for B, and that D is not in the belief state for C, yet these classifier labelings are uncorrected in the initial labeling. Without the Bayesian updating there is no way of knowing what the label *should* be; however one could at least make a guess and do better than simply leaving these as is. Therefore Figure 4(a) includes a third data set for the sake of comparison that indicates how well the initial labeler would work if these labels were changed. In particular, whenever a case was found to have tasks labeled as B and F or C and D, a random guess was made from the two viable options, chosen from the weighted distribution based on the probabilities of each being the case. The updated labeler still clearly outperforms this labeling, with statistically significant results at all levels.

To make the test a bit more interesting, we added more uncertainty to the situation by examining performance when the accuracy of labeling B and C was reduced to 75%. Results are depicted in Figure 4(b). Note that we did not include a third case for "guessing", since the uncertainty in the B or C labeling means that one cannot be sure whether D or F are not viable options. Results show that the updated labeler performs well (with statistically significant results) up to 75% labeling accuracy. Performance drops off at this point, since the method

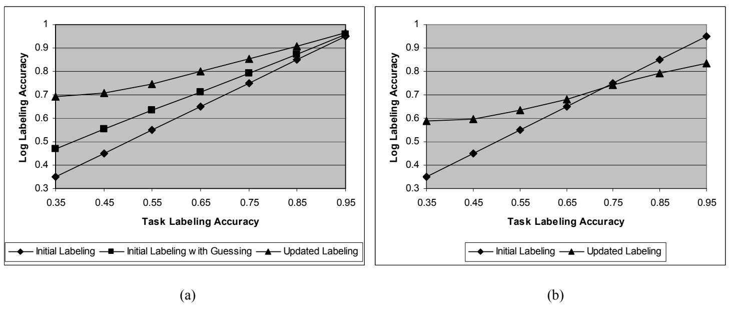

Fig. 4. Results for the two experiments

relies on identifying the belief state to make a decision, which relies on whether B or C is executed, which is only determined correctly 75% of the time. This demonstrates the need for the iterative process described in section 3. Optimal accuracy cannot be obtained in one step, but must instead be obtained by making small increases in different parts of the workflow through a number of iterations.

6 Conclusions and Related Work

In this paper, we present a technique for refining the labeling of tasks in a transaction log by analyzing properties of the workflow model mined up to a certain point in an iterative process. The resulting higher task labeling accuracy subsequently yields more accurate workflows. The technique works by taking an initial labeling obtained perhaps via machine learning-related techniques, such as classification or clustering, and iteratively applying Bayesian updates to the beliefs as to an event's true task label. We examine how the updates would be computed when two different types of information is obtained: (1) a suggested label and (2) a probability distribution over the set of labels. Experimentation on a simple case showed that, even when accuracy is high, the updater still could improve accuracy, and when the labeler's accuracy was low, updating more than doubled the accuracy. We also verified empirically that updating is effective even when there are errors in the workflow model being used to refine the belief state.

A great deal of work has been done in the area of workflow mining in recent years, particularly by Aalst et al [3] and Cook and Wolf [6]. However, relatively little has been done in the area of iterative or progressive construction of workflow models, although van der Aalst [1] discusses the notion of workflow extension from an a-priori model. Moreover, most workflow mining research found in the literature assumes that task labels are accurate and readily available, which is not necessarily the case in most practical contexts. Bayesian updating has been applied previously in many different areas. In particular, Buffett and Spencer [5] have applied Bayesian updating to the classification of opponent preference relations in the area of automated negotiation.

7 Future Work

The main focus for future work will be further development of the iterative workflow mining technique. Now that a method is in place for updating beliefs for the task labelings and resulting workflows in a given iteration, we are free to explore more deeply the effectiveness of the idea of iterative workflow mining, as well as the associated new research problems that are sure to arise in the investigation.

References

1. W.M.P. van der Aalst. Process mining and monitoring processes and services: Workshop report. In *F. Leymann, W. Reisig, S.R. Thatte, and W.M.P. van der Aalst, editors, The Role of Business Processes in Service Oriented Architectures, number 6291 in Dagstuhl Seminar Proceedings*, 2006.
2. 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.
3. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
4. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proceedings of the 6th International Conference on Extending Database Technology*, pages 469–483, 1998.
5. S. Buffett and B. Spencer. A bayesian classifier for learning opponents preferences in multi-object automated negotiation. *Electronic Commerce Research and Applications*, 6(3):274–284, 2007.
6. J.E. Cook and A.L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. In *ACM Trans. Softw. Eng. Methodol*, pages 147–176, 1999.
7. G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining frequent instances on workflows. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 209–221, 2003.
8. N. Kushmerick and T.A. Lau. Automated email activity management: An unsupervised learning approach. In *Proceedings of the 2005 International Conference on Intelligent User Interfaces*, pages 67–74, 2005.
9. James L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977.
10. A. Rozinat and W.M.P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *Proc. of the First International Workshop on Business Process Intelligence (BPI'05)*, pages 1–12, 2005.
11. R. Silva, J. Zhang, and J.G. Shanahan. Probabilistic workflow mining. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–284, 2005.
12. H.M.W. Verbeek, A.J. Pretorius, W.M.P. van der Aalst, and J.J. van Wijk. On petri-net synthesis and attribute-based visualization. In *Proceedings of the Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 127–142, 2007.