Using Classification Methods to Label Tasks in Process Mining

Buffett, Scott; Geng, Liqiang

Publisher's version / Version de l'éditeur:
https://doi.org/10.1002/smr.463
Journal of Software Maintenance and Evolution : Research and Practice, 22, 6-7, pp. 497-517, 2010-09-01

NRC Publications Record / Notice d'Archives des publications de CNRC:
https://nrc-publications.canada.ca/eng/view/object/?id=2bfe8ee0-7c36-4851-ac10-7617a1d39a55
https://publications-cnrc.canada.ca/fra/voir/objet/?id=2bfe8ee0-7c36-4851-ac10-7617a1d39a55

Using Classification Methods to Label Tasks in Process Mining

Scott Buffett and Liqiang Geng

Institute for Information Technology - e-Business, National Research Council,

Fredericton, New Brunswick, Canada, E3B 9W4

{scott.buffett, liqiang.geng}@nrc.gc.ca

Abstract. We investigate a method designed to improve the accuracy of process mining in scenarios where the identification of task labels for log events is uncertain. Such situations are prevalent in business processes where events consist of communications between people, such as email messages. We examine how the accuracy of an independent task identifier, such as a classification or clustering engine, can be improved by examining the currently mined process model. First, a classification scheme based on identifying keywords in each message is presented to provide an initial labeling. We then demonstrate how these labels can be refined by considering the likelihood that the event represents a particular task as obtained via an analysis of the current representation of the process model. This process is then repeated a number of times until the model is sufficiently refined. Results show that both keyword classification and current process model analysis can be significantly effective on their own, and when combined have the potential to correct virtually all errors when noise is low (less than 20%), and can reduce the error rate by about 85% when noise is in the 30-40% range.

Keywords: 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 process mining. Process mining involves automatically (or semi-automatically) inspecting 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 identify interesting patterns such as common sequences of events or common communications between parties. This information can then be used to determine how the process can be made more efficient, or can instead be compared with ongoing activity to ensure employees are complying with the common workflow. To accomplish this effectively, there are a number of interesting problems that have been investigated in the recent literature. These include algorithms for discovering causal relations in activities and complex constructs [3, 4], efficient methods for analyzing large logs [3], user-friendly visualization of discovered workflows [26], and conformance analysis between discovered models and observed activity [6, 24], among others.

One thing that much of the literature has failed to address, however, is 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. Consider, for example, attempting to model the process of conducting travel planning within a large company. Many of the tasks involved in such a process may take the form of communication between two parties, perhaps in the form of an email message. In this case, we may want to analyze such messages and attempt to discover where they fit within the process being modeled. However, determining what task a particular message represents can be quite difficult. 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 employee? Each of the two events 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 [20]. 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". Such a message could be mistakenly labeled as an itinerary confirmation.

A possible solution to this problem is then to examine the currently constructed process model as an independent source of information on the likelihood of labels. Consider again the travel confirmation scenario, where the event was erroneously classified as an itinerary confirmation message. One might be able to look at typical flows in the process 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.

In this paper, we present task labeling solutions utilizing each of keyword classification and current process model analysis, and propose a mechanism for considering the two together to obtain a result that is superior to using either individually. This will provide a more accurate labeling of tasks, which will, in turn, result in a more accurately mined process model. The techniques demonstrated here utilize a Bayesian approach. Initially, a naive Bayesian classifier is used to determine a prior task labeling over the set of events examined in the log, and a process model is built based on this initial labeling. Each event is then examined again and, by considering both the trace of events in the case in which the event appears as well as the current process model, a probability distribution is constructed over the event's possible positions in the model, yielding a set of probabilities over the possible tasks. The two types of information are then

[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.

used together to determine a more accurate prediction of the task that the event represents, and the event is possibly given a new label. Finally, a process model is mined using the new labels. We develop and test techniques for use in two different scenarios: (1) where the keyword classification and current process model analysis can be integrated to provide a single mechanism for determining label probabilities, and (2) where the two technologies are implemented separately and each provide a probability distribution over the set of task labels for each event, which are then integrated using further Bayesian analysis to determine the posterior probabilities.

One problem that remains here is that the process model upon which label probabilities are determined is mined using inaccurately labeled tasks. To overcome this deficiency, we employ an iterative approach to the problem. As long as the initial label classification has a degree of accuracy that is sufficient to ensure that the updates based on the model will make some progress and improve the process model, more improvement is likely to be observed each time the process is repeated. This is because of the fact that each step will use a more accurate labeling (and thus a more accurate model) than the previous. In our experimentation, we assess the effectiveness of the technique for varying levels of initial accuracy, and also study the effectiveness of varying numbers of iteration rounds.

One should note that, while we present our material in the context of analyzing text-based indicators for tasks, the techniques demonstrated will be applicable in any situation where task labels are not known but rather a number of clues used for predicting the task are present. In the text-based scenario, such clues are simply the set of keywords that are mined from the accompanying body of text. However, other examples of where techniques can be applied include predicting patients' next location in a hospital according to their ages, genders, diseases, symptoms, as well as their previous trajectories in the hospital to increase the bed turnaround rate. One should also note that, while we consider that task labels are uncertain and can only be probabilistically predicted, in this work we assume that the instance or case to which the event corresponds is known with certainty. We address case uncertainty in a future work.

This paper builds on prior research [5] presented at the Fourth Workshop on Business Process Intelligence (BPI 08), which was held in conjunction with Business Process Management (BPM 2008) in Milan, Italy. This previous work focused solely on the computational technique for refining a probabilistic labeling from a classifier by incorporating new likelihoods from the current workflow analysis. We extend that work here by demonstrating the entire iterative process within which the refinement incrementally takes place, by discussing the inner workings of the keyword classifier, by simplifying and generalizing the previously used refinement computation, and by vastly broadening the scope of experimentation and analysis. To assist in the reader's comprehension and appreciation of the ideas, we have also added a running example. The paper is organized as follows. In section 2 we offer a thorough review of the literature, and introduce two important concepts, namely process mining and Bayesian classification. We also discuss the task labeling problem in more detail, and outline the research goals of the current work presented in this paper. Section 3 discusses the naïve Bayes classification and current workflow analysis methods for probabilistically determining labels for log events, and section 4 demonstrates how information from these two independent sources can be incorporated to determine a superior labeling, thus yielding a more accurate workflow model. Section 5 offers extensive experimentation, results and analysis, while we conclude the paper in section 6 with some final thoughts, as well as some plans for future work.

2 Background

2.1 Literature Review

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 a number of practical contexts.

Our work in this paper involves both content analysis (i.e. text mining) and link analysis (i.e., workflow analysis). The past decade has seen much work conducted in field of workflow mining. The most investigated problem is to create graph based workflow models from structured logs, where activities have already been correctly recorded or identified (see van der Aalst *et al.* [2] for a good survey). Our work differs from the traditional workflow mining in that we assume that we already have an initial workflow model and we want to refine it based on additional (or external) evidence.

Other related work includes topic identification using text mining techniques. Identifying topics from emails is a popular problem. Clustering is the most frequently used method for email topic identification. Huang et al. proposed to use clustering methods to infer activities from emails based on the subject and body of the emails [17]. This is one of the first works in this domain. Li *et al.* incorporated semantic analysis for email clustering [22]. The method identifies the entities such as person and date in the subjects, and uses these entity names and sentence patterns together with the keywords in subjects as the features for clustering. Classification is another method for identifying topics in emails. Dredze *et al.* proposed a method to classify emails into activities based on the people involved in the activities and the content of the email messages [8]. They defined similarity measures based on the sets of senders and the recipients to calculate the similarity values between an email and a topic. These measures were then used in the email classification process. Other techniques to discovery topics from emails include latent semantic analysis and latent Dirichlet allocation models [9], and formal concept analysis [11]. However, in these techniques, topics in emails are solely determined by the content and subjects, and correlations between emails are not taken into consideration.

Recently some researchers have started to pay attention to both content-based methods and link-based methods to decide the labels for events embodied in emails and other artifacts. Kushmerick and

Lau tried to identify the workflow from an email dataset of e-commerce transactions [21]. The learning process of their method consists of three steps. First, activities of each transaction are identified using the identifiers like transaction numbers. Secondly, transitions of the processes are identified using the hierarchical agglomerative clustering method. Thirdly, the workflow model, represented by an automaton, is derived. To our knowledge, this is the first work that derives a workflow model from emails. However, it only deals with a specific case in e-commerce transactions, where the emails are automatically generated according to templates.

Similar work has been done to infer speech acts and links between the speech acts from emails. Khoussainov and Kushmerick combined relation identification and speech act classification to improve the performance of email topic identification [18]. They first identify the relations among emails and the speech acts for each email. Then they repeatedly use relations of emails to reassign speech acts to emails and use speech acts as features to update the relations among emails. This iterative process refines the classification results. However, no work has been done to predict activities according to an existing workflow model and the evidence intrinsic to the event, and to update such labels and models iteratively.

2.2 Process Mining

Process mining, also referred to as 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 represents 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, such as directed acyclic graphs [4], finite state machines [6], variations of Bayesian networks [25], workflow schemas [13], and Petri Nets [2, 23]. 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. This example indicates that any sequence where A is executed, followed by both B and either C or D (in parallel), followed by E is legal. By examining the accompanying log and inspecting any sequence of events that correspond to the same case number, one can confirm that this specified workflow is always the case.

(*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)

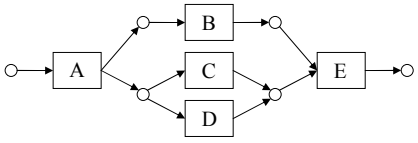

Fig. 1. Example log and corresponding workflow diagram.

While van der Aalst is responsible for a large body of work in process mining (see e.g. [2, 3]), more recently Duster *et al.* developed a tool that can take logs from a process-aware collaboration system like Caramba and extract relevant information for ad-hoc process mining [10]. Greco *et al.* proposed an automatic process mining approach to discover a taxonomical model that represents process models at

different abstraction levels. In their approach, a preliminary schema decomposition is first discovered by using a divisive clustering algorithm. Then the schema is restructured into a taxonomy by generalizing all the different schemas in the corresponding subtree [14]. Ghionna *et al.* proposed an clustering based method to identify workflow outliers. This method takes into consideration both the log files and the process models [12].

2.3 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 $\frac{0.4 \times 0.05}{0.44} = 0.05$.

A naïve Bayesian classifier is a simplified Bayesian model in that it assumes all attributes are independent from each other, and therefore it is very efficient to train a naïve Bayesian classifier. Despite its simplicity, empirical studies show that the predictive accuracy of the naïve Bayesian classifier is very similar to that of other complex classification models, such as the Support Vector Machine [16]. Zhang et al. conducted theoretical studies to explain the apparently unreasonable efficacy of naïve Bayesian classifiers [27]. Recently naïve Bayesian classification has been widely used in text mining problems [7, 15, 19].

2.4 Goals of the Paper

In order to demonstrate the main focus of the paper and corresponding research goals that are addressed, we briefly introduce the task labeling problem and sketch the procedure we propose for improving accuracy when predicting the task represented by a given log event. To accomplish this, we take the reader through a simple example. Consider the event log in Figure 2. The figure also contains a Petri net representing the underlying workflow of the tasks in the log. Compare this workflow representation with the Petri net constructed for the log in Figure 3, where perhaps labels are uncertain and must be predicted. The table representing the log in this figure contains an extra column indicating the predicted label for each event. Note that the "true" (unknown) labels are the same as those in Figure 2, and thus the underlying true workflows should be the same. However, the predicted labels in Figure 3 contain two errors, namely for events 5 and 33 (in bold). Even though this represents only a $2/40 = 5\%$ error rate, the process model that

is then mined using the predicted labels more than doubles in size in terms of the number of transitions (including dummy transitions), and also introduces a complex workflow component, in this case a loop. So we can observe that even a highly accurate (but imperfect) labeler is still likely to produce highly inaccurate process models, which allows us to conclude that even small improvements in the label accuracy will result in substantial improvements to the workflow in terms of the number of invalid firing sequences that are eliminated.

Event	*Case*	*Task*	*Event*	*Case*	*Task*
1	1	A	21	3	C
2	6	A	22	9	C
3	1	B	23	9	D
4	2	A	24	2	D
5	5	A	25	4	D
6	3	A	26	7	C
7	7	A	27	3	B
8	6	B	28	8	B
9	1	C	29	2	E
10	8	A	30	5	C
11	8	D	31	7	B
12	4	A	32	8	E
13	1	E	33	4	B
14	9	A	34	7	E
15	2	B	35	3	E
16	9	B	36	10	B
17	6	D	37	4	E
18	5	B	38	5	E
19	6	E	39	10	D
20	10	A	40	10	E

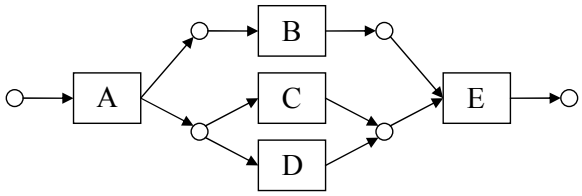

Fig. 2. Example log and corresponding workflow diagram.

Event	*Case*	*Task*	*Label*	*Event*	*Case*	*Task*	*Label*
1	1	A	A	21	3	C	C
2	6	A	A	22	9	C	C
3	1	B	B	23	9	D	D
4	2	A	A	24	2	D	D
5	5	A	**B**	25	4	D	D
6	3	A	A	26	7	C	C
7	7	A	A	27	3	B	B
8	6	B	B	28	8	B	B
9	1	C	C	29	2	E	E
10	8	A	A	30	5	C	C
11	8	D	D	31	7	B	B
12	4	A	A	32	8	E	E
13	1	E	E	33	4	B	**C**
14	9	A	A	34	7	E	E
15	2	B	B	35	3	E	E
16	9	B	B	36	10	B	B
17	6	D	D	37	4	E	E
18	5	B	B	38	5	E	E
19	6	E	E	39	10	D	D
20	10	A	A	40	10	E	E

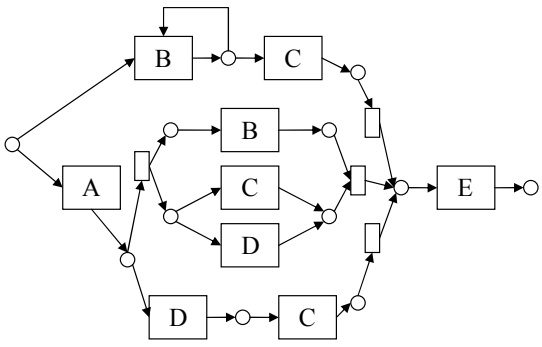

Fig. 3. Example log with predicted labels and corresponding erroneous workflow diagram.

An obvious approach to solving this problem is then to look at the typical flows of activity, and simply eliminate sequences that are highly infrequent from the process model. If label errors are infrequent, then this approach should be successful. However, there is the danger that there are indeed valid traces that are infrequently observed and will thus not be included in the model. This might be acceptable in some applications of workflow mining. However, in others where perhaps the goal is to identify inappropriate or suspicious activity by comparing observed activity with the accepted model of the process, it is just these very traces that are needed in order to distinguish between unexpected activity and activity that is quite acceptable but just not very common. Thus a better approach is needed.

Our proposed technique considers both the confidence of the task classifier as well as the likelihood of observing the activity in the current context of the business process to determine whether or not to attempt a label correction. For example, if the classifier had very low confidence that event 5 was a "B" (perhaps believing with only 55% confidence that the event was a "B" as opposed to 45% for "A"), and the process model indicated that observing a "B" was highly unlikely in this position, and an "A" would be much more probable, then we could perhaps choose to re-label. However, if the classifier assigned a high probability to "B", then we would likely prefer to leave it unchanged.

We then have two independent sources of information on which to base task label predictions, where the two sources may possibly conflict. To accomplish the ultimate end goal of considering all information, determining a posterior probability distribution over the set of tasks for each event in the log, and finally choosing the most likely task label, we address the following questions in this paper:

1. How is classification performed on the raw events to obtain an initial labeling?
2. How are task label beliefs computed using the current process model, and, considering the complexity of obtaining a fully accurate belief state, how can we determine task probabilities quickly in a realistic setting?
3. How do we integrate the classification procedure with the process model analysis?

4. If integrating the above two procedures is not an option, how can we combine the output of the classifier with the belief state from the process model analysis to determine a posterior probability distribution over the set of tasks?
5. How can we perform the process iteratively to incrementally refine the model?

3 Computing Task Label Beliefs

In this section we discuss two independent methods for determining the likelihood that an event should be labeled by a particular task, the naïve Bayes classification approach and the current process analysis approach. In the following section we demonstrate how to incorporate the two.

3.1 Naïve Bayes Classification for Task Labeling

For a specific workflow, the number of possible activity types is fixed. Classification algorithms can then be used to train a model on the keywords in the log files to classify the events into activities. A naïve Bayesian classifier is a probabilistic classifier based on Bayes' theorem with the assumption that all features are independent of each other given a class. We chose to use naïve Bayesian classification to identify tasks (classes) based on several considerations. First, it is an efficient algorithm compared with other classifiers. Secondly, it is easy to incorporate new features, which makes it more flexible. Thirdly, it was reported in many applications that its accuracy is satisfactory in spite of its independence assumption.

The process for identifying activities from the content of emails and other artefacts is as follows. Stop words which convey no helpful information for classification are removed from the text. Then all other words are stemmed to their root form. Next, the keywords are selected using feature selection methods, such as information gain or term frequency-inverse document frequency. Then each keyword is considered as a feature and a table is constructed showing whether a keyword occurred in an event, or how many times a keyword occurred in the event. This table is then used as the training data for the classifier.

According to the naïve Bayesian classification method, given a set of keywords $w_1, w_2, \ldots w_k$ for an event, the probability that the event belongs to an activity A can be defined as

$$P(A|w_1, \ldots, w_k) \propto P(A) \prod_{i=1}^{k} P(w_i|A) \tag{2}$$

$P(A)$ and $P(w_i|A)$ can be obtained directly from training data. We assign the event to the activity label with the maximum posterior probability.

$$A = \arg\max_i P(A_i|w_1, \ldots, w_k) \tag{3}$$

To demonstrate the technique, we commence our running example. Consider the partial event log in Figure 4. The first 20 rows contain the training set, indicating the keywords found for each event, as well as the task label of the event, which is assumed to be correct. The final row of the table shows the keywords found for an event for which the correct label is unknown. Note that there may be several other keywords found for each event in the training set. However, for determining the label of the new event, only the keywords found for the new event are relevant, so for clarity only those are shown for the training events. The probability that each label should be used for the unknown event is computed using equation 2. Only probabilities for D, E and F are shown, as probabilities for all other labels are equal to 0:

$w_1 =$ "*confirm*", $w_2 =$ "*itinerary*":

$$P(D) = 0.1, P(E) = 0.1, P(F) = 0.05$$

$$p(D|w_1) = 0.25, P(D|w_2) = 0.4$$

$$p(E|w_1) = 0.25, P(E|w_2) = 0.2$$

$$p(F|w_1) = 0.125, P(F|w_2) = 0.2$$

Given the keyword evidence $Ev = w_1, w_2$,

Keywords	Label
	A
	B
confirm, itinerary	D
	A
itinerary	B
	A
confirm	E
	A
confirm	C
confirm	G
confirm, itinerary	D
confirm	A
	G
	G
	C
	C
confirm, itinerary	E
confirm, itinerary	F
	G
	G
⋮	⋮
confirm, itinerary	???

Fig. 4. Training data for label classification

$$P(D|Ev) \propto 0.1 \cdot 0.25 \cdot 0.4 = 0.010$$

$$P(E|Ev) \propto 0.1 \cdot 0.25 \cdot 0.2 = 0.005$$

$$P(F|Ev) \propto 0.05 \cdot 0.125 \cdot 0.2 = 0.001$$

Normalizing each value by the sum 0.016 gives:

$$P(D|Ev) = 0.625$$

$$P(E|Ev) = 0.313$$

$$P(F|Ev) = 0.063$$

and thus "D" is chosen as the label.

3.2 Using the Current Process Model to Determine Task Probabilities

The process of determining the probability distribution over the set of possible tasks for an event in the context of the process model is performed by constructing a *belief state* for the event. A belief state essentially indicates (1) the likelihood of various incarnations of the process model being the true model, and (2) given any such hypothetical model, the likelihood of the event residing in any location in the model. The belief state is determined based on the beliefs regarding the task labelings over the set of log events, and is updated after each step of the iterative labeling process. Based on these beliefs, one can then compute the probability of an event representing a given task. Formally, let C be the set of cases in the log, let T be the set of task labels and let $\mathcal{W}$ be the set of possible workflow models. In the initial iteration, the set of workflows could be one or more that are initially manually specified, a workflow that is mined from a training set of cases $C' \subset C$, or simply be the workflow mined using the uncertain labels provided by the classifier for entire set C of cases. Next, let L_c be the set of possible labelings for case $c \in C$, where

each $\ell_c \in L_c$ maps each event in c to a member of the set T of possible tasks. These different possible labelings could be those provided by the classification engine. The possible labelings in L_c for each case then induces a set L of possible labelings over the entire log. The set $\mathcal{W}$ of possible workflows then consists of those workflows mined using each labeling in L.

The belief state $B(\mathcal{W}, x)$ for a given log event x then gives an indication of the possible locations in each workflow that the event x is likely to reside, based on probabilistic information obtained via mined workflows from the logs. In other words, it answers the question "Given that x is being executed, at what step are we in the business process?" 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 can then be used to determine the probability that x represents a particular task.

To demonstrate, we return to the running example from the previous section. Consider an event log in which the following seven cases are found:

Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
A	A	A	A	A	A	A
B	B	C	C	C	C	C
D	E	E	F	E	E	D*
G	G	G	G	G	G	G

Let x be the event labeled by "D*" from the example in the last section with possible task labels D, E and F. Then there are 3 possible labelings $L_7 = \{ACDG, ACEG, ACFG\}$ for case 7 (assuming A, C and G are known for sure). If x is the only uncertain event, then these labelings (along with those for cases 1-6) would result in the set of two possible workflow models $\mathcal{W} = \{W_1, W_2\}$ in Figure 5. Specifically, if x is labeled as "D", then W_1 will be the mined workflow model, while if x is labeled as "E" or "F", then W_2 will be the mined workflow model.

The belief state $B(\mathcal{W}, x)$ for x is then computed as a probabilistic labeling over the transitions residing in workflow models in $\mathcal{W}$ using workflow and task frequencies mined from the log. Looking at case 7, we see that event x follows C and precedes G. Examining all cases where such a workflow sequence can take place (i.e. cases 3-7), we see that possible locations for the transition representing x are (1) "D" in W_1, with probability 0.2 (since there is a 0.2 chance that both x is a "D" and W_1 is the workflow), (2) "E" in W_2, with probability 0.6, or (3) "F" in W_2, with probability 0.2. These probabilistic labelings (seen in Figure 5) make up the belief state.

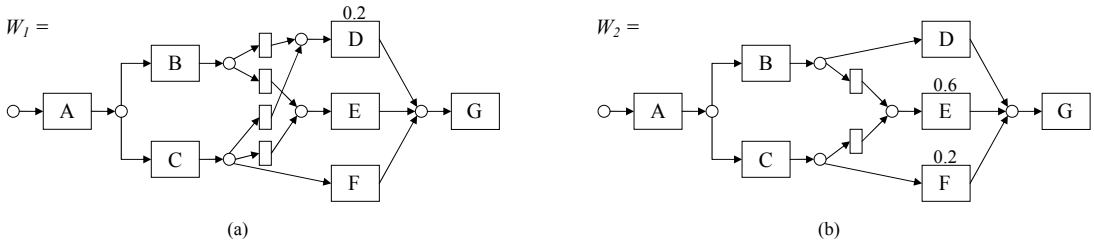

Fig. 5. Example Petri nets showing the belief state for an event x that is believed to follow task C and precede task G.

As a result, there are now two sets of independent probabilities for the labels for x. It is the goal of the next section to demonstrate how to incorporate the two sources of information to best determine how each task should be labeled, ultimately enabling us to select the best workflow model.

4 Incorporating Keyword Classification and Process Model Analysis

4.1 The Refinement Step

In this section we discuss how the two information sources for label probabilities are used together to produce a posterior probability distribution. We consider two scenarios: a naïve Bayes approach for use by the classification engine, and a decoupled approach which can be used by a secondary mechanism that

has no access to information used by the classifier, but rather only the probabilities over labels that it produces as a result.

The Integrated Approach with Naïve Bayes. Integrating information on the belief state for an event within the classification process consists of simply treating the belief state information as new evidence for the classifier. Let $w_1, \ldots, w_k$ be the keyword-based classifier evidence, and let Ev_w represent the workflow-based evidence. Then the probability of "A" being the event is computed using

$$P(A|w_1, \ldots, w_k, Ev_w) \propto P(A) \prod_{i=1}^{k} P(w_i|A) \cdot P(Ev_w|A) \tag{4}$$

In the example, if "CxG" is used as the evidence (i.e. that x is preceded by "C" and followed by "G"), then (referring back to the seven cases) $P(Ev_w|D) = 1/2 = 0.5$, $P(Ev_w|E) = 3/4 = 0.75$ and $P(Ev_w|F) = 1$. We find the new task label using

$$P(D|Ev) \propto 0.1 \cdot 0.25 \cdot 0.4 \cdot 0.5 = 0.005$$
$$P(E|Ev) \propto 0.1 \cdot 0.25 \cdot 0.2 \cdot 0.75 = 0.004$$
$$P(F|Ev) \propto 0.05 \cdot 0.125 \cdot 0.2 \cdot 1 = 0.001$$

Giving new probabilities

$$P(D|Ev) = 0.5$$
$$P(E|Ev) = 0.4$$
$$P(F|Ev) = 0.1$$

"D" remains chosen the most likely label, however the likelihood has dropped somewhat, meaning this new evidence casts some doubt on whether the event actually represents a "D".

The Decoupled Approach. The problem becomes more complex when integrating information from the workflow into the classification process is not an option, perhaps when external techniques or software are being used to perform the initial classification, and only the output from the classifier in the form of a probability distribution over the set of possible tasks is obtained. There are now two independent probabilities associated with each label, that obtained from the classifier as well as that obtained via the belief state. The focus here is to present a method for updating the classifier probabilities with the belief state probabilities to compute a posterior probability distribution.

Let $P(t|Ev)$ be the probability that an event x should be labeled as task t given the keyword evidence Ev found by the classification engine, given that $P(t)$ is the probability of a random event in the log being t. Given a belief state $B(\mathcal{W}, x)$, let $P'(t)$ be the new likelihood that an event is actually a t. The goal is to compute the new posterior probability $P'(t|Ev)$. To give some intuition for what this really means, the problem can be framed as follows: "If the probability of x representing task t, given the keyword evidence Ev, is $P(t|Ev)$ when t represents $P(t)$ percent of the population, what would be the new probability $P'(t|Ev)$ if $P(t)$ was changed to $P'(t)$?" We derive the formula as follows. Consider Bayes' rule:

$$P(t|Ev) = \frac{P(t)P(Ev|t)}{P(Ev)} \tag{5}$$

Using the updated $P'(t)$ gives the new probability $P'(t|Ev)$ (while also affecting $P(Ev)$ but not $P(Ev|t)$):

$$P'(t|Ev) = \frac{P'(t)P(Ev|t)}{P'(Ev)} \tag{6}$$

Rearranging equation (5) we get

$$P(Ev|t) = \frac{P(t|Ev)P(Ev)}{P(t)} \tag{7}$$

And substituting (7) into (6)

$$P'(t|Ev) = \frac{P'(t)P(t|Ev)P(Ev)}{P(t)P'(Ev)} \tag{8}$$

Ignoring $\frac{P(Ev)}{P'(Ev)}$, which is constant across all labels, $P'(t|Ev)$ can be computed using

$$P'(t|Ev) \propto \frac{P'(t)P(t|Ev)}{P(t)} \tag{9}$$

To demonstrate, we return to the running example. Let the belief state dictate that $P'(D) = 0.2$, $P'(E) = 0.6$ and $P'(F) = 0.2$ (i.e. the probability of observing D, E and F in cases 3-7). We find the new task label using

$$P'(D|Ev) \propto 0.2 \cdot 0.625/0.1 = 1.25$$
$$P'(E|Ev) \propto 0.6 \cdot 0.313/0.1 = 1.88$$
$$P'(F|Ev) \propto 0.2 \cdot 0.063/0.05 = 0.25$$

Normalizing to give probabilities

$$P'(D|Ev) = 0.37$$
$$P'(E|Ev) = 0.56$$
$$P'(F|Ev) = 0.07$$

Thus, due to the strong new evidence suggesting that "E" should be the label, combined with the classifier's relative weak confidence in "D", the new information has caused us to change the label from "D" to "E".

4.2 Iterative Refinement Process

Because of the complexities involved in computing the belief state, the refinement process may need to be iterated a number of times in order to reach an equilibrium state. This is due to the fact that belief state

for an event could be determined based on incorrect labelings for other events in the log. Consider the example discussed in the last section. The belief state for x was constructed based on the knowledge that x followed "C" and was followed by "G". Had there been errors in the "C" and "G" labelings, however, the belief state would have been inaccurate. However, with a number of iterations, belief states should become more and more accurate, which will cause more accurate task labelings, and vice-versa.

In each iteration, belief states are updated, labels are modified where suggested, and a refined set $\mathcal{W}$ of possible workflow models is produced as a result. The new workflow models are then used to produce the belief states in the next iteration, and so on. 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 process model.

5 Results

In this section, we demonstrate the effectiveness of our methods by performing a number of experiments. First we test each of the belief model prediction and the keyword classification methods individually, to show that each performs well on their own. We then examine the performance of our two methods for combining the two prediction techniques, and finally show how increased accuracy is realized by iterating the process a number of times. Note that we define labeling accuracy as the number of correctly labeled tasks in an event log, divided by the number of events in the log.

5.1 Refining Labels Using the Belief State

We demonstrate the effectiveness of the label probability updates by presenting a few results for a simple example. We used a simulated log file containing 1000 cases, where each case was one of $ABDG$, $ABEG$, $ACEG$ or $ACFG$, appearing with various frequencies, yielding the workflow model demonstrated by the Petri net in Figure 6. 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. We did not use actual classification in this case, as we wanted to ensure that these results were entirely independent of our classifier's performance. We bring the two together later. 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. Then, based on whether $ABxG$ or $ACxG$ was observed, the belief state was computed and the most likely label selected.

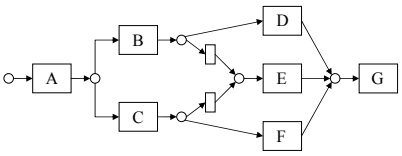

Fig. 6. Process model used in the experiments

Figure 7(a) compares the accuracy of the initial labeling with the updated labeling over the 1000 log cases, given different levels of accuracy for the initial classifier labeling. 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 (since there are three possibilities). Note that all results are statistically significant at the 0.05 level.

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%. Uncertainty in the correctness of B and C will lead to erroneous belief states, which are to be expected in practice. Results in Figure 7(b) 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 relies on identifying the belief state, which

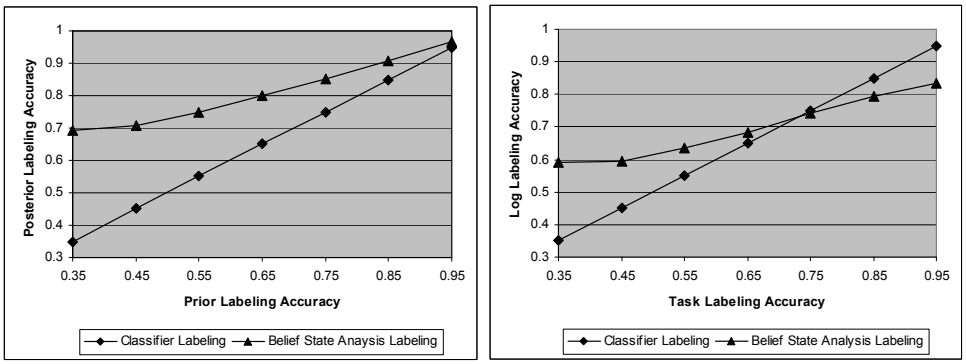

Fig. 7. Results for the two experiments, demonstrating the increased accuracy in the posterior labeling after considering the belief state, when the belief state is known with (a) 100% and (b) 75% certainty.

is only reliable 75% of the time. This demonstrates the need for the iterative process. 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.

5.2 Keyword Classification

We independently tested the performance of the naïve Bayes classification method by defining a fictional process model for a travel planning scenario, and simulating a number of corresponding email messages that represent the various tasks in the business process. Figure 8 depicts the workflow used in our fictional process[2].

To generate the data, we developed a computer operation monitoring tool to record email messages sent and received in MS Outlook. The authors of this paper took the roles of employee, group leader, administrative assistant, travel agent, etc., in the workflow[3]. We then extracted keywords from these email

[2] Note that this process is largely based on the actual travel planning process employed at the National Research Council in Canada. Acknowledgements go to Louise-Ann Trainor for her consultation in the development of the model.

[3] Thanks to Hongyu Liu and Yonghua You for participating in these roles.

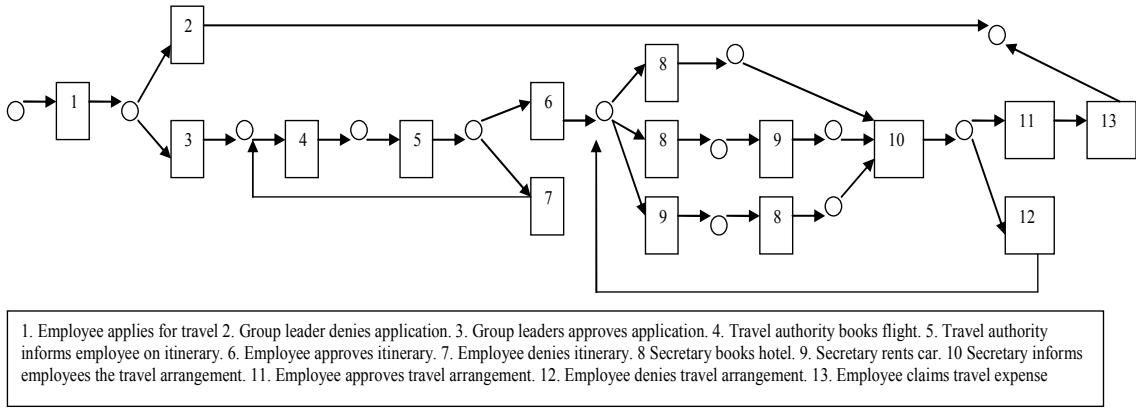

Fig. 8. Workflow diagram for a fictional travel planning process

messages based on the frequency of occurrences, and simulated a number of new messages, resulting in a large corpus of data.

The simulated email messages with different levels of noise were generated in two steps. First, structured event logs were generated from the workflow models. Then, the keywords associated with each event and the timestamps were generated. We generated ten audit trail data sets with noise levels ranging between 0% and 90%, in increments of 10%. We considered three types of noise: insertion of random words from a dictionary, deletion of keywords, and replacement of keywords with other words in the dictionary. The three types of noise were added with the same probability. Each audit trail data set contained 100 instances with around 1400 events.

We then examined the email messages, identified the keywords, and used naive Bayes classification to determine the labels. Figure 9 depicts the results for each noise level. The classifier performed quite well, maintaining accuracy levels greater than 90% whenever noise was less than 40%, before dropping off more sharply as noise was increased.

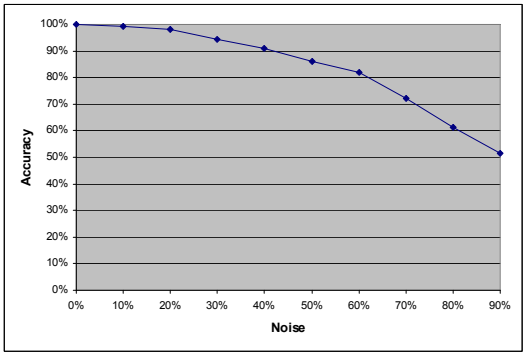

Fig. 9. Accuracy of keyword classifier given various levels of noise

5.3 Integrating Keyword Classification with Belief State Analysis

Next we used the integrated approach described in section 4.1 to combine each of the two prediction methods, using the data from the travel planning scenario described in the previous section, and iterated through the refinement process a number of times. Figure 10(a) depicts the results observed when noise was less than or equal to 40%. In each case, the error rate was reduced dramatically after one iteration, with a substantial improvement after the second pass, reducing the error rate by 85% or more. Progress is maintained in each subsequent round, however it appears that no significant improvement is made after round 2. For the sake of completion, we separately depict the results for noise 50% and higher in Figure 10(b). As expected, when noise is sufficiently high the error rate will worsen with each pass, since the labels are refined using increasingly inaccurate belief states.

5.4 Decoupled Classification and Belief State Analysis

Finally we analyze performance of the method that considers output from keyword classification and belief state analysis separately, once again on the travel planning data described previously, with charts depicted in Figure 11. Belief states were determined using a simple method of only observing the preceding and following tasks, and determining the likelihood of each task appearing between them. For example, for the

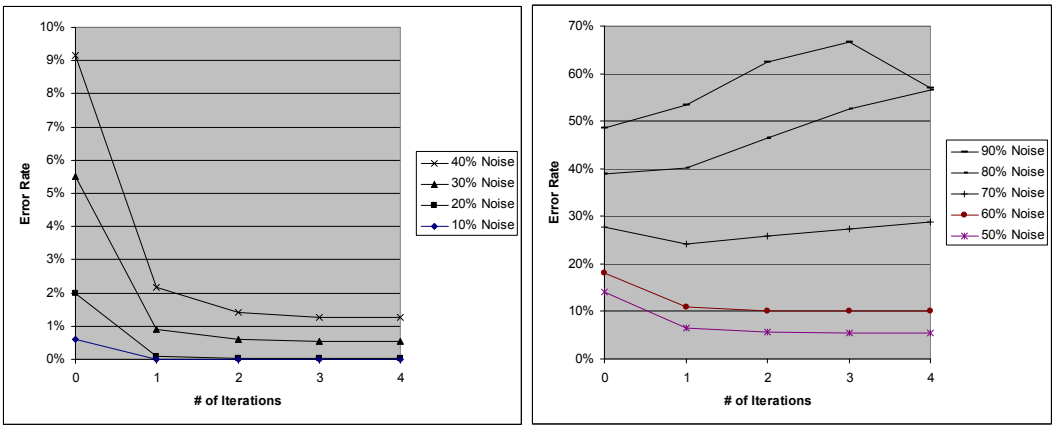

Fig. 10. Results for the integrated approach with (a) low levels of noise and (b) high levels of noise. Note that results for round "0" indicate initial performance of keyword classification without any refinement from the belief state probabilities

execution trace ...AxC... where x is the log event in question, the probability that x = B is computed by dividing the number of occurrences of ABC in the log by the number of occurrences of AyC where y is any event. We defer the problem of more accurate belief state estimation to future work.

While the results are inferior to those of the integrated approach for low levels of noise, performance is still impressive. Most importantly, there is evidence that significant improvement can still be made in the error rate by using this approach when integrated label refinement is not an option. For higher levels of noise this technique does not tend to deteriorate as rapidly as the integrated approach, since the corresponding inaccurate belief states tend to not be given as much weight due to the decreased confidence in the probabilities. As we continue to develop new ways for constructing more accurate belief states, we are sure to achieve even better performance.

6 Conclusions and Future Work

In this paper, we present a technique for labeling tasks in a transaction log by considering associated clues (in our case, keywords in some sort of text-based message) to perform classification on events to

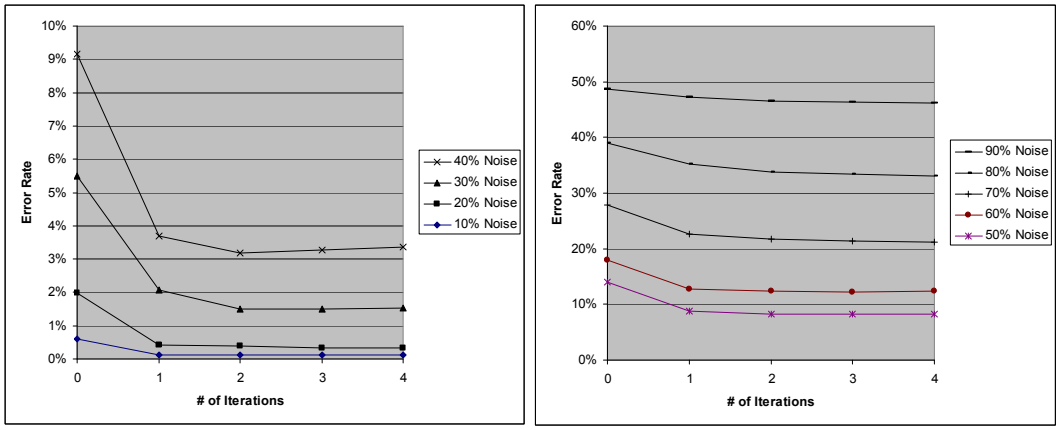

Fig. 11. Results for the decoupled approach with (a) low levels of noise and (b) high levels of noise.

predict their task labels, and refine those labels by analyzing properties of the currently mined process model. The resulting higher task labeling accuracy subsequently yields more accurate process models. The technique works by taking the initial labeling obtained via the keyword classification engine, and iteratively incorporates probabilistic information over the set of possible labels obtained by determining a belief state in the process model for a given event.

Results show that both keyword classification and belief state-based labeling can be somewhat effective on their own, and when combined have the potential to correct virtually all errors when noise is low (less than 20%), and can reduce the error rate by about 85% when noise is in the 30-40% range. We also examined the effectiveness of repeating the process in a number of iterations and found that, while the technology does continue to make progress at each step, no significant progress seems to be realized after round number 2.

The main focus for future work will be further development of techniques for accurately determining the belief state. In particular, a mechanism for determining a number of different candidate workflows is needed, along with an accompanying method for determining the probability of each of those process models

actually representing the true model. Based on this set of models, a technique to accurately determine the likelihood of an event residing in each location is required as well.

We also plan to implement this technology in a practical setting, namely the area of ensuring compliance in the handling of private information within an enterprise.

Finally, we plan to lift our assumption that case labelings are known for certain. We are currently investigating the use of clustering techniques to solve this problem.

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 L. Geng. Bayesian classification of events for task labeling using workflow models. In *Proceedings of the 4th Workshop on Business Process Intelligence (BPI 08)*, Milan, Italy, 2008.
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. W. Dai, G. Xue, Q. Yang, and Y. Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI2007)*, pages 540–545, 2007.

8. M. Dredze, T.A. Lau, and N. Kushmerick. Automatically classifying emails into activities. In *Proceedings of the 2006 International Conference on Intelligent User Interfaces*, pages 70–77, 2006.

9. M. Dredze, H. M. Wallach, D. Puller, and F. Pereira. Generating summary keywords for emails using topics. In *Proceedings of the 2008 International Conference on Intelligent User Interfaces*, pages 199–206, 2008.

10. Schahram Dustdar, Thomas Hoffmann, and Wil M. P. van der Aalst. Mining of ad-hoc business processes with teamlog. *Data Knowl. Eng.*, 55(2):129–158, 2005.

11. L. Geng, L. Korba, Y. Wang, X. Wang, and Y. You. Finding topics in email using formal concept analysis and fuzzy membership functions. In *Proceedings of Canadian Conference on AI*, pages 108–113, Windsor, ON, Canada, 2008.

12. Lucantonio Ghionna, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Outlier detection techniques for process mining applications. In *ISMIS 2008*, pages 150–159, 2008.

13. 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.

14. Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Mining taxonomies of process models. *Data Knowl. Eng.*, 67(1):74–102, 2008.

15. F. He and X. Ding. Improving naive bayes text classifier using smoothing methods. In *Proceedings of the 29th European Conference on IR Research (ECIR)*, pages 703–707, 2007.

16. J. Huang, J. Lu, and C. X. Ling. Comparing naive bayes, decision trees, and svm with auc and accuracy. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 553–556, 2003.

17. Y. Huang, D. Govindaraju, T.M. Mitchell, V.R. de Carvalho, and W.W. Cohen. Inferring ongoing activities of workstation users by clustering email. In *Proceedings of the First Conference on Email and Anti-Spam*, Mountain View, California, USA, 2004.

18. R. Khoussainov and N. Kushmerick. Email task management: An iterative relational learning approach. In *Second Conference on Email and Anti-Spam (CEAS)*, Stanford University, California, USA, 2005.

19. S. Kim, K. Han, H. Rim, and S. Myaeng. Some effective techniques for naive bayes text classification. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 18(11):1457–1466, 2006.

20. 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.

21. 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, San Diego, California, 2006.

22. H. Li, D. Shen, B. Zhang, A. Chen, and Q. Yang. Adding semantics to email clustering. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 938–942, Hong Kong, China, 2006.

23. James L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977.

24. 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.

25. 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.

26. 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.

27. H. Zhang. Exploring conditions for the optimality of naïve bayes. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 19(2):183–198, 2005.