Function-based Case Classification for Improving Business Process Mining

Yaguang Sun and Bernhard Bauer
Programming Distributed Systems Lab, University of Augsburg, Augsburg, Germany

Keywords: Business Process Mining, Multi-label Case Classification, Sequential Pattern Mining, Business Process Extension.

Abstract: In the last years business process mining has become a wide research area. However, existing process mining techniques encounter challenges while dealing with event logs stemming from highly flexible environments because such logs contain a large amount of different behaviors. As a result, inaccurate and wrong analysis results might be obtained. In this paper we propose a case (a case is an instance of the business process) classification technique which is able to combine domain experts knowledge for classifying cases so that each group is calculated containing the cases with similar behaviors. By applying existing process mining techniques on the cases for each group, more meaningful and accurate analysis results can be obtained.

1 INTRODUCTION

Business process mining techniques aim at analysing various aspects of enterprise business processes such as workflow discovery, process performance analysis and organizational structure analysis, etc. The starting point of these analyses is usually an event log which is a set of cases, where each case is an instance of the business process (van der Aalst, 2011). Cases always have an attribute *trace* which is a finite sequence of ordered events. In an event log, events and cases are represented by unique identifiers (event ID and case ID). Except for trace, a case may also have other attributes such as originator and cost, etc. An event can also have its own attributes. Table 1 shows an example event log.

Table 1: An example event log.

Case id	Event id	Properties		
		Activity	Resource	Cost
1	421	A	Pete	40
	422	B	Sun	200
	423	C	Simon	300
	424	D	Chris	100
	425	E	Pete	200
2	452	A	Mike	30
	453	C	Simon	300
	454	F	Chris	200
	455	D	Sun	100
	456	G	Mike	500
...	...	...	...	...

In the real world business processes are often executed in highly flexible environments, e.g., healthcare, customer relationship management (CRM) and product development (Weerdt et al., 2013). The event logs that stem from such flexible environments (real-life event logs) often contain a high variety of case behaviors. The behaviors of a certain case are reflected by the values of case attributes. For instance, the behavior of a case related to the case attribute trace is called structural case behavior which is expressed by the events and their precedence relations in the trace. Cases are generated so as to adhere to a certain category of behaviors determined by business process-based domain criterion (Weerdt et al., 2013). The process mining techniques are facing difficulties while dealing with event logs containing multiple process-based criteria. For instance, inaccurate and complex business process models might be generated by existing workflow discovery algorithms with an input of such logs because multiple criteria may represent a lot of structural behaviors of cases from structurally different process executions. Furthermore, some important process analysis results related to certain process criterion might be concealed in the final results generated by using the logs containing multiple process criteria. There is a need to classify the cases with different behaviors into different groups.

Accordingly, some approaches are developed for

Sun Y. and Bauer B..
Function-based Case Classification for Improving Business Process Mining.
DOI: 10.5220/0005349202510258
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 251-258
ISBN: 978-989-758-096-3

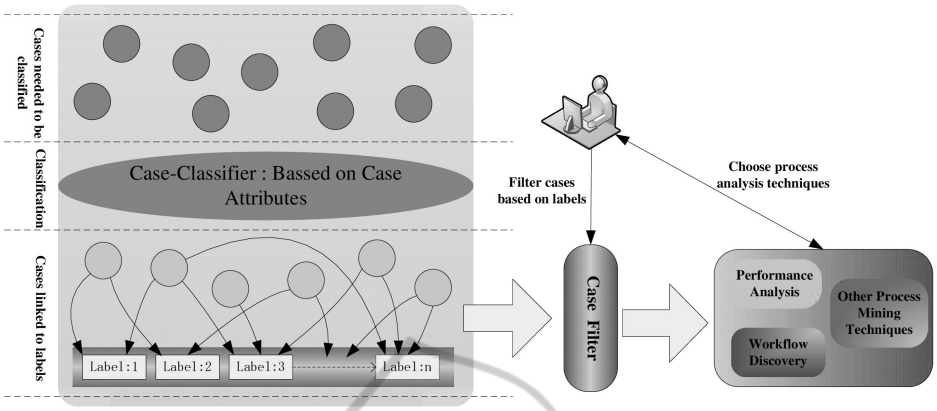


Figure 1: Model of case classification in the scenario of process mining.

this purpose. One efficient technique is *trace clustering* (Greco et al., 2006; Song et al., 2009; Bose and van der Aalst, 2010; Bose and van der Aalst, 2009) which is designed mainly to help discover better workflows. It is able to automatically capture the behaviors of cases in an event log and then group the cases with similar behaviors into the same sub-log. By applying process mining techniques on each simpler sub-log more accurate and understandable analysis results can be obtained.

Trace clustering can help find a lot of hidden behaviors among the cases. However, it is an unsupervised learning technique and lack domain knowledge. As a result it is unable to indicate which behaviors found are crucial for classifying the cases or which behaviors are wanted by customers for classifying the cases. Additionally, treating all of the behaviors found equally may not generate a correct or meaningful classification of cases.

Classification (supervised learning) which is able to combine the domain knowledge from business experts can be a useful tool for classifying cases. In this paper we put forward a case classification technique:

- We demonstrate and formalise the problem of *multi-label case classification* in Section 3.1.
- One important sub-problem of the case classification problem is how to exploit the case attribute trace for classifying the cases. To solve this sub-problem we develop a systematic method based on *sequential pattern mining* technique for utilising the case attribute *trace* for classifying cases in Section 3.2 and Section 3.3.
- To test the efficiency of our method, we make a case study in Section 4 by using a real-life event log of a Dutch academic hospital from Business Process Intelligence Contest 2011 (BPIC 2011).

2 PROBLEM DESCRIPTION

Figure 1 depicts a detailed model for case classification in the scenario of business process mining. Basically, the case-classifier links the primitive cases in an event log with the labels. Each label represents one category and cases connected to the same label may share some common behaviors. Finally by working with the already classified cases, the process mining techniques are able to analyse the enterprise business process in different points of view and generate more readable and meaningful analysis results.

Traditional data classification approaches proceed in two steps (Kotsiantis, 2007):

- In the **first step**, training data is analysed by a classification algorithm so that a function $y = f(X)$ can be learned. The generated function (classifier) is able to predict the associated category label y of a given tuple $X = (x_1, x_2, \ldots, x_n)$, where X represents the set of attributes of a specific item.
- In the **second step**, predictive accuracy of the classifier built is estimated by utilising a test set made up of tuples and their relevant category labels.

A *training event log* should be generated firstly for a *case classification problem*. Labels are added manually to part of the cases in an event log by domain experts according to their domain knowledge and the behaviors of cases. Then these labeled cases are extracted to generate the *training event log* and *test event log*. A *training event log* is used for building the classifier and a *test event log* for estimating the performance of the classifier. By looking into some real-life event logs we discover that a case may have more than one label which makes this case classifica-

tion problem a *multi-label classification* problem (as shown in Figure 1).

The behaviors of cases are reflected by the values of case attributes recorded in the event log. Most case attributes that have discrete values or numeric values can be easily utilised for building the classifier or judging which labels a case belongs to. But the *trace* of a case can not be directly used. A trace is an important attribute of a case which is a finite sequence of ordered events, for instance, in Table 1 the trace of $case1$ is $< A,B,C,D,E >$. The trace may be a major element for deciding which labels a case pertains to (while the labels are related to the structural feature of trace).

For the cases generated by structured business processes[1], it is easy to transform the traces into a suitable form that can be utilised as an attribute for case classification because the compositions of traces are limited by a structured process model. However, for the cases from a real-life event log, such a transformation should not be directly carried out because it is possible that cases with similar features may seem very different from each other. Additionally, there might be multiple structural features (presence or absence of an activity, presence or absence of combinations of activities and so on) of traces related to one label. So how to capture the possible structural behaviors of traces (expressed by the events and their precedence relations in the traces) relevant to labels is an important sub-problem for solving the *multi-label case classification problem*.

To solve the *multi-label case classification problem* and its sub-problem mentioned above, we proposed in this paper a *sequential pattern mining* technique-based method (Section 3) for mining all of the possible label-related structural features of traces and then transforming these found features into suitable forms of case attributes to help the later case classification.

3 BASIC CONCEPTS AND APPROACH DESIGN

In this section we first elaborate the concept of *multi-label case classification* (Section 3.1). Afterwards the structural feature of traces is presented in a formal way (Section 3.2). At last we provide a method for transforming the found structural features of traces into a suitable form of case attributes for the later case classification (Section 3.3).

[1]A structured business process is a rigidly defined process with a model which considers all of the process instance permutations and every process instance complies with this model. If a structured business process has a loop structure it can also generate a large amount of isomerous traces.

3.1 Multi-label Case Classification

A multi-label classification technique solves the problem of predicting to which set of classes (also represented by labels) a new instance belongs by exploiting a training set of data. The training data is a set $T = \{t_1, t_2, \ldots, t_n\}$ of already classified samples where each sample t_i is constructed by a k-dimensional vector $X_{t_i} = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$. The dimensions in X_{t_i} represent attributes of the sample, as well as the categories to which t_i pertains.

The existing multi-label classification methods are mainly divided into two types, one is algorithm independent and the other one is algorithm dependent (Tsoumakas and Katakis, 2007; Carvalho and Freitas, 2009). In this paper we will utilise the algorithm independent approach for solving the problem of multi-label case classification. In the algorithm independent approach, a multi-label classification problem can be converted into several single-label problems. For each label (or category) a classifier is built so that the classification problem related to this label can be dealt with. For the multi-label case classification problem in this paper, the training data is conveyed by the following definitions:

Definition 1. Let C_t be the set of training cases. A case $c \in C_t$ is defined as a tuple $c = (N_c, L_c, \Theta_c)$, where $N_c = \{n_1, n_2, \ldots, n_k\}$ is the set of names of case attributes, L_c is a set of labels, $\Theta_c : N_c \rightarrow A_c$ is an attribute-transition function which maps the name of an attribute into the value of this attribute, where A_c is the set of attribute values for case c. A label $l \in L_c$ represents a manually given class to which case c belongs.

As already mentioned, a case in an event log may be assigned multiple labels in the real world, thus for all $c \in C_t$, we have $|L_c| \geq 1$, where $|L_c|$ is the number of labels.

Definition 2. A training event log is defined as $E_t \subseteq C_t$, for any $c_1, c_2 \in E_t$ such that $c_1 \neq c_2$.

Let's presume that the example event log in Table 1 is a training event log, all cases in this log have an attribute *originator* and an attribute *labels*, $case1$ has an originator "*Mike*" and a set of assigned labels $\{l_1, l_2\}$. According to the concepts defined above, $\Theta_1(originator) =$ "*Mike*" is the originator for $case1$, $\Theta_1(trace) =< A,B,C,D,E >$ is the trace for $case1$, $L_1 = \{l_1, l_2\}$ is the set of labels for $case1$.

Definition 3. The multi-label case classification problem is defined as $Prob = (SE_t, SE_{test}, \Phi, \Xi)$, where SE_t is the set of training event logs, SE_{test} represents the set of test event logs for evaluating the accuracy of the learned classifiers, Φ is a multi-label classification algorithm, Ξ is a classifier evaluation schema. $\Phi : SE_t \rightarrow \Psi$ represents the process for building a classifier (or a set of classifiers), where Ψ is the set of classifiers. $\Xi : SE_{test}, \Psi \rightarrow \{Accurate, Inaccurate\}$ represents the process for evaluating the performance of a built classifier.

3.2 Definitions Relevant to Function

The *sequential pattern mining* techniques solve the problem of finding all frequent subsequences from a given set of sequences, where each sequence contains a list of ordered events and each event consists of a set of items (Han and Kamber, 2000). A minimum support threshold is manually given for judging if the occurrence of a subsequence is frequent or not.

Let $I = \{I_1, I_2, \ldots, I_n\}$ be the set of all items, $S = \{S_1, S_2, \ldots, S_m\}$ be the set of all sequences. A sequence $S_i \in S$ is an ordered list of events and denoted $< e_{i1}, e_{i2}, \ldots, e_{ij} >$, where each event e_{ik} represents an item set composed by items of I. For any two sequences $\alpha =< a_1, a_2, \ldots, a_l >$ and $\beta =< b_1, b_2, \ldots, b_q >$ over S, α is a subsequence of β, if $1 \leq p_1 < p_2 < \cdots < p_l \leq q$ such that $a_1 = b_{p_1}, a_2 = b_{p_2}, \ldots, a_l = b_{p_l}$. Let $D \subseteq S$ be a database of sequences, for a given minimum support threshold min_sup ($0 < min\_sup < 1$), a sequence λ is called a sequential pattern if $support(\lambda) \geq min\_sup \times |D|$, where $support(\lambda)$ is the number of sequences in D which contain λ and $|D|$ represents the total number of sequences in D.

As introduced in Section 3.1, the trace of a case is a sequence of ordered events. Thus the set of traces collected from an event log can be deemed as a sequence database on which the sequential pattern mining algorithms can be implemented. We define a sequential pattern mined from a set of traces extracted from an event log as a *function*:

Definition 4. Let E be an event log, $Y \subseteq E$ be a set of traces collected from E, $\Gamma : Y \overset{min\_sup}{\longrightarrow} F$ be a sequential pattern mining algorithm, where $F = \{f_1, f_2, \ldots, f_n\}$ is the mined set of sequential patterns with a minimum support threshold min_sup, a sequential pattern $f_i \in F$ is defined as a *function* relevant to Y and F is also called a set of *functions*.

Definition 5. Let E_t be a training event log, $Y_{label} \subseteq E_t$ be a set of traces from E_t, where each trace in Y_{label} is related to one common label. A label-related function set F_{label} is a set of functions mined from Y_{label}, and a function $f_k \in F_{label}$ is called a label-related *function*.

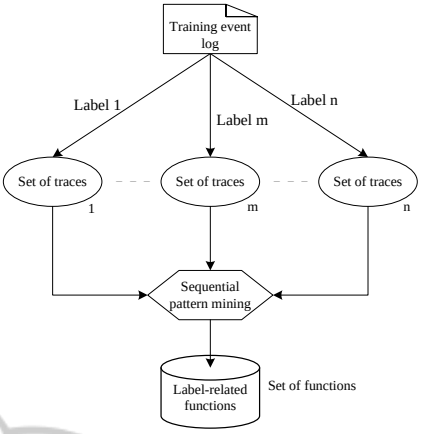


Figure 2: Mining label-related functions from a training event log.

In our opinion, the label-related *functions* in F_{label} reveal the commonly and frequently appeared structures of the traces in Y_{label}. As mentioned in Section 2, a label may be associated with the structural characteristics of traces. While this is true, the label-related functions can be exploited to judge if a trace belongs to a specific label.

Figure 2 illustrates the process of mining all of the possible label-related *functions* from a training event log. In the first step, all the traces included by a training event log are separated into different sets where each set is associated with one label. For example, in Figure 2 the traces of cases with label *1* are collected and sent to set *1*. In the second step, a sequential pattern mining procedure is executed on each set of traces for discovering label-related *functions*. Finally all of the found *functions* are grouped together in one set (label-related *function* set).

Let's presume that the event log shown in Table 1 is a training event log and each case in this log has a set of labels from $\{l_1, l_2, l_3, l_4\}$. For this training event log, a set of traces Y_{l_2} is obtained through extracting all traces with label l_2. Then by mining Y_{l_2} using a sequential pattern mining method a label-related *function* set F_{l_2} can be extracted. This label-related function mining procedure mentioned above is described in Algorithm 1.

3.3 Transforming Label-related Functions into Case Attributes

The raw label-related functions found could not be exploited directly, they should be transformed into a suitable form of case attributes so that they can be

Algorithm 1 Mining label-related functions for a given training event log

Input: a training event log E_t, the minimum support threshold *min_sup*
Let $G(label)$ be a set of traces related to a specific label in E_t.
Let S be a set of all possible label-related functions for E_t.
Let LB be the set of labels in E_t.
Let Θ be a case attribute-transition function as described in Definition 1.
Let $\Gamma(Y, m\_s) \rightarrow F$ be a sequential pattern mining algorithm, where Y is a sequence database, m_s is a minimum support threshold, F is the mined set of sequential patterns.

$S \leftarrow \emptyset$ // is set to be an empty set
for each $G(label_i)$ such that $label_i \in LB$ **do**
  $G(label_i) \leftarrow \emptyset$
end for
for each case $c_j \in E_t$ **do**
  for each label $l_k \in L_{c_j}$ **do**
    $G(l_k) = G(l_k) \cup \Theta_{c_j}(trace)$ // L_{c_j} is the set of labels owned by c_j
  end for
end for
for each label $lb \in LB$ **do**
  $S = S \cup \Gamma(G(lb), min\_sup)$
end for

Output: the set of all label-related functions in E_t: S

checked by a classification algorithm. In the following parts we will propose a method for this purpose.

Through algorithm 1, the traces in a training event log are grouped into different sets where each set is related to one label and the *functions* for each set are discovered. Before building a classifier for each label, the found *functions* need to be converted into usable case attributes. To explain our method clearly, we will first set up a few variants: E_t is a training event log, a set $G = \{G(label_1), G(label_2), \ldots, G(label_n)\}$ where each $G(label_i)$ is a set of all the traces relevant to $label_i$ in E_t, a set $F = \{F(label_1), F(label_2), \ldots, F(label_n)\}$ where each $F(label_j)$ is a function set for $G(label_j)$ found by algorithm 1, $F^* = \{function_1, function_2, \ldots, function_m\}$ is a set which contains all of the functions in F. An association table AT is then established which connects each function in F^* with a global unique identifier. In the association table shown in Table 2, for instance, $function_1$ has an id A_1.

Table 2: An example association table for the functions in F^*.

Function ID	Function
A_1	$function_1$
A_2	$function_2$
$\vdots$	$\vdots$
A_n	$function_n$

Let $c_p \in E_t$ be a case from training event log E_t, add all of the function ids in the association table as attribute names to the attribute list of c_p, and their initial values are set to 0. The next step is to calculate the value of each new added attribute. Take function A_1 from Table 2 as an example, a subsequence-detection process is carried out with $\Theta_{c_p}(trace)$ (the trace of c_p) and $function_1$ (matched with A_1 in Table 2) as inputs. If $function_1$ is judged to be a subsequence of $\Theta_{c_p}(trace)$, then $\Theta_{c_p}(A_1)$ is reset to be 1. The procedure mentioned above should also be applied to both the test event logs and the normal event logs which contain cases needed to be classified.

4 CASE STUDY

We tested the effectiveness of our technique on the hospital event log from the BPIC 2011. This log contains 624 activities and 1143 cases where each case stands for a treatment process of a patient of the gynaecology department. A lot of attributes relevant to the cases have been recorded in this log, such as the ages of patients and the final diagnosis for patients.

In the experiment the treatments performed on a patient are regarded as labels (categories). One reason is that in healthcare industry treatment is often used as a label for classifying cases, for instance, the SAP Business Suite for Patient Management exploits the treatment as one category for case classification (SAP Community, 2014). There are overall 48 kinds of treatment (coded by number) in this log from which we have chosen seven frequently happened treatments (namely *13*, *23*, *61*, *101*, *113*, *603*, *3101*) for analysis. Each case may belong to more than one treatments and each treatment may be characterized by multiple behaviors of *traces*.

For generating the training event log, a simple strategy mentioned in (Tsoumakas and Katakis, 2007) is exploited which discards every multi-label case in the hospital event log. For instance, all the cases that have only a single treatment belonging to the treatment set $TS = \{13, 23, 61, 101, 113, 603, 3101\}$ are extracted and form the training event log. All of the

Table 3: Performances of the classifiers built for each treatment in *TS*.

Treatment	Correctly classified instances ratio	AUC	Kappa statistic	Recall
13	0.896940	0.935	0.6662	0.988/0.882
23	0.900161	0.850	0.6060	0.782/0.917
61	0.901771	0.968	0.7241	0.729/0.959
101	0.597424	0.679	0.2935	0.481/1.000
113	0.887279	0.740	0.0590	0.068/0.973
603	0.855072	0.758	0.3304	0.638/0.873
3101	0.853462	0.873	0.5335	0.900/0.847

multi-label cases are organized as the test event log. As a result, we obtain a training event log with 279 cases and a test event log with 621 cases.

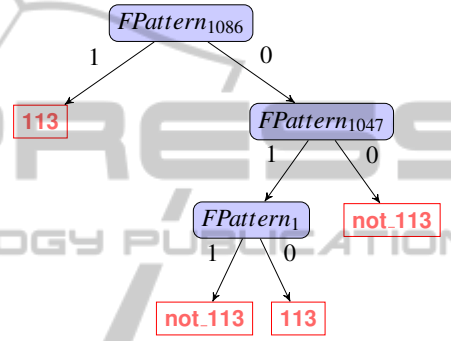
In Algorithm 1 the closed sequential pattern mining algorithm[2] *BIDE* (Wang and Han 2004) is used. By implementing Algorithm 1 with the training event log and a $min\_sup = 0.3$ as input parameters, a label-related function set S which contains 3159 functions has been discovered. Then all of the found functions are transformed into case attributes for both the training event log and the test event log through method introduced in Section 3.3. In the generated association table AT, the id of a function is in the form of $FPattern_k$ where k represents the position of this function in AT. For example, $A_1 = FPattern\_1$ in Table 2 because $function_1$ is the first item and $A_n = FPattern\_n$ because $function_n$ is the n^{th} item in this association table.

In this paper we utilise an algorithm independent approach for solving the problem of multi-label case classification which converts the learning problem into traditional single-label classification. For each element in *TS* a classifier is learned by using a training event log. For example, for treatment *13*, a classifier is built which is able to judge if a case falls in treatment *13* or not. In our experiment seven binary classifiers (because seven kinds of treatment are analysed) are established.

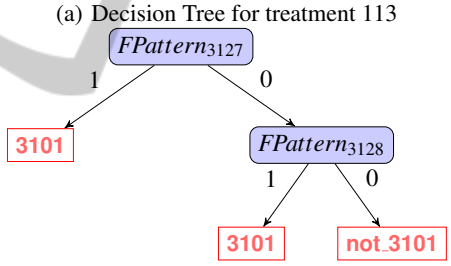
To testify the standpoint that the labels pertained by a case may be related to the structural feature of its trace, only the case attributes generated by the transformation of discovered label-related functions are considered in our experiment, for instance, in this hospital event log, the other case attributes such like *Age* and *Diagnisis* are not put to use.

Firstly we use the *Decision Tree*-based algorithm

[2]The reason to use a closed sequential pattern mining algorithm is that it effectively decreases the total number of sequential patterns generated but in the meantime preserves the complete information about all the sequential patterns.



(a) Decision Tree for treatment 113



(b) Decision Tree for treatment 3101

Figure 3: Decision Trees built for treatment 113 and treatment 3101.

C4.5 (Quinlan, 1993) in our experiment. For each treatment in *TS* a decision tree is built by exploiting the attributes of cases. For example, we got a decision tree for treatment *113* as shown in Figure 3(a), and Figure 3(b) shows the decision tree for treatment *3101*. According to the decision tree for treatment *113*, a case will be inferred to belong to treatment *113* if it has an attribute $FPattern\_1086 = 1$, and not if it has the attributes $FPattern\_1086 = 0$ and $FPattern\_1047 = 0$ at the same time.

Table 3 shows the performances for the seven classifiers evaluated by using the test event log generated. Several parameters are calculated in the evaluation step. *The area under the ROC curve* (AUC) which has a value between 0 and 1 reflects the performance of the classification model. An ideal classifier has an AUC value close to 1. *Correctly Classified*

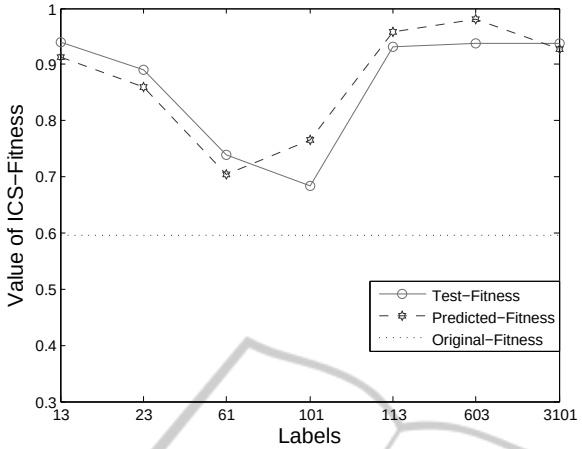


Figure 4: ICS-Fitness for the models generated by using the sub-logs from *SC* and *PC*.

Instances Ratio reflects the total classification accuracy of a specific classifier. The *Kappa Statistic* (KS) measures the diversity factor between the classification results from a classifier built and a classification by chance, $KS \in (0.75, 1)$ implies that the effect of the classifier is very good, $KS \in (0.4, 0.75)$ is characterized as fair to good and $KS \in (0, 0.4)$ as poor. *Recall* measures the proportion of correctly classified instances among all the instances with the same label or without a specific label.

Additionally, to testify the practicability of our technique in business process mining area, we then evaluate the effectiveness of the classification results (obtained by using Decision Tree algorithm) on the *process discovery* task (the most crucial learning task in process mining domain). Let C_{test} be the test event log generated in our experiment, $SC = \{C_{13}, C_{23}, C_{61}, C_{101}, C_{113}, C_{603}, C_{3101}\}$ be the set of sub-logs where each sub-log contains the cases correlates to one treatment from TS. Let $PC = \{PC_{13}, PC_{23}, PC_{61}, PC_{101}, PC_{113}, PC_{603}, PC_{3101}\}$ be the set of sub-logs where each consists of the cases predicted to have one same treatment from TS. Then process models for the entire test log C_{test} and for each sub-log in *SC* and *PC* are learned by using process discovery techniques. In our experiment *Heuristic Mining Algorithm* as described in (Weijters and Ribeiro, 2011) is utilised. Afterwards, the *ICS-Fitness* (Weerdt et al., 2013) (fitness measures the proportion of behavior in the event log possible according to the model) for each model is calculated and the results are shown in Figure 4. The ICS-Fitness related to each sub-log in *SC* is called *Test-Fitness* and the ICS-Fitness for the sub-logs in *PC* is called *Predicted-Fitness*. The *Original-Fitness* (as a base line in Figure 4) is calculated by using the entire test event log C_{test} and the model for it. In Figure 4 we can see that the Predicted-Fitnesses and Test-Fitnesses for most of the sub-logs in *SC* and *PC* are much higher than the Original-Fitness due to the sub-logs in *SC* and *PC* are simpler and contain less behaviors than the original event log C_{test}, as a result the models generated for these sub-logs become more accurate. By comparing the Predicted-Fitness with the Test-Fitness in Figure 4, we discovered that most of the Predicted-Fitnesses are very close to their relevant Test-Fitnesses, for instance, the value of Predicted-Fitness for PC_{3101} is 0.9267 and the value of Test-Fitness for C_{3101} is 0.9382 which is very close to the Predicted-Fitness for PC_{3101}. From the analyses mentioned above it can be deduced that the results from the multi-label case classification technique have some practical values in the business process mining area. *Process discovery* is only one perspective of business process mining techniques and the multi-label case classification method can also benefit other techniques in this area (e.g., business process performance analysis) to help generate more meaningful and accurate analysis results.

5 RELATED WORK

In the literature, different approaches have been put forward to overcome the negative impacts from high variety of behaviors stored in event logs on business process mining techniques. One efficient technique is *trace clustering*.

In (Song et al., 2009) the authors present an approach for characterizing the traces by profiles for the later trace clustering. Each profile is a set of items that describe the trace from a specific angle. Five profiles are defined in (Song et al., 2009), they are *activity profile*, *transition profile*, *case attributes pro-*

file, *event attributes profile* and *performance profile*. Then by converting the profiles defined into an aggregate vector the distance between any two traces can be calculated. One advantage of this technique is that it provides a full range of metrics for clustering traces.

Context-aware trace clustering methods are proposed in (Bose and van der Aalst, 2010) and (Bose and van der Aalst, 2009). In (Bose and van der Aalst, 2010) the authors indicate that the feature sets based on sub-sequences of different lengths are context-aware for the vector space model and can reveal some set of common functionality accessed by the process. Two traces that have a lot of common conserved features should be put in the same cluster. In (Bose and van der Aalst, 2009) the authors presents an edit distance-based approach for partitioning traces into clusters such that each cluster consists of traces with similar structure. The cost of edit operations is associated with the contexts of activities so that the calculated edit distance between traces is more accurate.

In (Weerdt et al., 2013) a novel technique for trace clustering is presented which is able to directly optimise the accuracy of each cluster's underlying process model. This method doesn't consider the vector space model or define a metric for trace clustering, it simply discovers the suitable traces for each cluster so that the combined accuracy of the related models for these clusters is maximized. This method sufficiently resolves the divergence between the clustering bias and the evaluation bias.

Classification technique is widely used on *Decision Mining* area in business process mining. In (Rozinat and van der Aalst, 2006) the author developed a *Decision Miner* based on *Decision Tree* algorithm which aims at analysing the choice constructs of process models by exploiting the event attributes recorded in event logs.

6 CONCLUSION

In this paper we proposed and elaborated the basic definition of *Multi-label Case Classification*. Next, a concrete systematic method was introduced which is able to discover all of the label-related structural features of traces and transform these found features into case attributes for the later classification job. The effectiveness and practicability of our technique were then testified through a case study.

Our next research task will be to focus on exploiting the decision trees generated by our technique so as to clearly reveal the influences of different categories on the execution of business processes.

REFERENCES

Bose, R. and van der Aalst, W. (2009). Context Aware Trace Clustering: Towards Improving Process Mining Results. In *Proceedings of the SIAM International Conference on Data Mining*, pages 401–412.

Bose, R. and van der Aalst, W. (2010). Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In *Business Process Management Workshops*, volume 43 of *Lecture Notes in Business Information Processing*, pages 170–181. Springer Berlin.

SAP Community. (2014). Customer-defined Case Classification. http://help.sap.com.

Greco, G., Guzzo, A., Pontieri, L., and Sacca, D. (2006). Discovering Expressive Process Models by Clustering Log Traces. *IEEE Transaction on Knowledge and Data Engineering*, 18(8):1010–1027.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2nd edition.

Kotsiantis, S. (2007). Supervised Machine Learning: A Review of Classification Techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24. IOS Press.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rozinat, A. and van der Aalst, W. (2006). Decision Mining in Prom. In *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer Berlin.

Song, M., Gnther, C., and van der Aalst, W. (2009). Trace Clustering in Process Mining. In *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 109–120. Springer Berlin.

Tsoumakas, G. and Katakis, I. (2007). Multi-label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3(3):10–13.

Carvalho, A. and Freitas, A. A. (2009). A Tutorial on Multi-label Classification Techniques. *Foundations of Computational Intelligence, Studies in Computational Intelligence*, pages 117-195. Springer Berlin.

van der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Berlin Heidelberg, Berlin, 1nd edition.

Weerdt, J. D., vanden Broucke, S., Vanthienen, J., and Baesens, B. (2013). Active Trace Clustering for Improved Process Discovery. *IEEE Transaction on Knowledge and Data Engineering*, 25(12):2708–2720.

Weijters, A. and Ribeiro, J. (2011). Flexible Heuristics Miner (FHM). In *Proceedings of CIDM*, pages 310–317.

Wang, J. and Han, J. (2004). BIDE: Efficient Mining of Frequent Closed Sequences. *In 20th Int. Conf. on Data Engineering*, Boston, MA.