Discovering Structured Event Logs from Unstructured Audit Trails for Workflow Mining

Geng, Liqiang; Buffett, Scott; Hamilton, Bruce; Wang, Xin; Korba, Larry; Liu, Hongyu; Wang, Yunli

Publisher’s version / Version de l'éditeur:
https://doi.org/10.1007/978-3-642-04125-9_47
Proceedings of the 18th International Symposium on Foundations of Intelligent Systems(ISMIS '09), 2009-09-17

Discovering Structured Event Logs from Unstructured Audit Trails for Workflow Mining

Liqiang Geng[1], Scott Buffett[1], Bruce Hamilton[1], Xin Wang[2], Larry Korba[1],
Hongyu Liu[1], and Yunli Wang[1]

[1] IIT, National Research Council of Canada, Fredericton, Canada, E3B 9W4
[2]Department of Geomatics Engineering, University of Calgary, Canada, T2N 1N4
{liqiang.geng, scott.buffett, bruce.hamilton}@nrc.gc.ca, xcwang@ucalgary.ca,
{larry.korba, hongyu.liu, yunlin.wang}@nrc.gc.ca

Abstract. Workflow mining aims to find graph-based process models based on activities, emails, and various event logs recorded in computer systems. Current workflow mining techniques mainly deal with well-structured and -symbolized event logs. In most real applications where workflow management software tools are not installed, these structured and symbolized logs are not available. Instead, the artifacts of daily computer operations may be readily available. In this paper, we propose a method to map these artifacts and content-based logs to structured logs so as to bridge the gap between the unstructured logs of real life situations and the status quo of workflow mining techniques. Our method consists of two tasks: discovering workflow instances and activity types. We use a clustering method to tackle the first task and a classification method to tackle the second. We propose a method to combine these two tasks to improve the performance of two as a whole. Experimental results on simulated data show the effectiveness of our method.

1 Introduction

Workflow mining refers to the task that automatically finds business process models within an enterprise by analyzing the computer operations, usually in the form of event logs, by a group of people involved in the process. These process models are usually represented in graphs and can be used to reengineer the work process and to ensure that the employees comply with the standard procedures.

Currently, most of the techniques for workflow mining require that structured (symbolized) event logs are available from certain software tools, such as Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) software, or Workflow Management systems [1]. However, in most real situations, these systems may not be installed in the enterprises. Therefore, structured event logs are not readily available. Instead, what we can have in most situations is the unstructured logs related to activities, such as emails sent and received, web pages accessed, documents edited, and applications executed. This kind of information can be easily obtained from web, email and application servers. These unstructured logs do not record the purpose of the operations, i.e., activity labels for the operations, and the labels for the workflow instances that these operations belong to. For example, an email message recorded in an unstructured log may contain keywords “trip”, “application”, “July”, “London”, “Richard”, and “Smith”. We neither know the corresponding activity type for this email, nor do we know the process instance to which the email belongs from the email itself. Unfortunately, current workflow mining techniques based on the structured logs cannot be used in such situations [1]. In this paper, we propose a method to identify the activity label and process instance for each event in the unstructured logs based on the keywords and named entities identified from the content of the event, and people involved in the operation. In this way the unstructured logs can be converted to structured logs, and hence can be used as the input to the workflow mining algorithms. For example, the task of the email message described above may be labeled with activity label *Trip-Application* and process instance label *Smith-London-Trip*.

The past decade has seen much work conducted in the field of workflow mining. The most investigated problem is to create graph-based workflow models from structured logs. A number of different graphic representations have been used for workflow mining. These include directed acyclic graph [2], finite state machine [5], variation of Bayesian network [9], workflow schema [6], and Petri Net [1]. Based on these representations, various algorithms have been proposed. However, all of the above-mentioned work is based on structured logs where both activity types and process instances are recorded.

Recently some researchers have started to pay attention to the content-based methods which utilize data mining techniques to decide whether to label two events as the same activity type or as part of the same workflow instance. Kushmerick and Lau used data mining methods to identify activities and transitions of activities from email messages [8]. However, that work is focused on solving a very specialized problem in E-commerce transactions. Also, they treat the task of indentifying activities and that of identifying workflow instances separately. Khoussainov and Kushmerick [7] reported a method to identify links and relations between email messages and the email speech acts [4]. However, their claim that integration of link identification and speech act identification will improve the performance for each other is based on two small data sets. In our paper, we performed systematic experiments on synthetic data sets and found that the combination of the two tasks of our problem does not necessarily improve the performance for each other.

The contributions of this paper include the following:

(1) We proposed a novel method that uses the *transition matrix* and *preceding matrix* to combine the results of the activity identification and workflow instance identification. This method takes into account the keywords, named entities, as well as the sequence information embodied in the unstructured logs.

(2) Our work is the first in this field based on the systematic experiments on synthetic data sets. In this way, we can see how data itself can affect the performance.

(3) We obtained observations that combining the instance identification and activity identification would not necessarily improve the results for each other, which is contrary to the claims made in [7]. We found that the quality of the data and the results of initial identification of the activities and workflow instances play a key role in the final results.

In section 2, we introduce basic concepts and state the problem we will tackle. In Section 3, we present our method to combine activity identification and instance identification tasks. In Section 4, we present the design of experiments and the experimental results. Section 5 concludes the paper.

2 Preliminaries

In this paper, we refer to the unstructured logs as *audit trails* and the structured logs as *event logs*. Audit trails and event logs are defined as follows.

An ***audit trail entry*** *ate* is a 5-tuple (*Op*, *SO*, *Rec*, *Cont*, *TS*), where *Op* refers to the operation type, *SO* refers to the operators, *Rec* refers to the recipients if applicable, *Cont* refers to the content of the artifacts related to the operation, and *TS* refers to the time stamp. An ***audit trail*** *AT* is a set of audit trail entries ranked by timestamp in an ascending order.

Given a set of activity types *A*, an ***event*** *e* is a 3-tuple (*Ins*, *Act*, *TS*), where *Ins* is an integer referring to the workflow instance label. $Act \in A$ is an activity type. *TS* is the time stamp. An ***event log*** *EL* is a set of events ranked by the timestamp in an ascending order.

In some literature, *activity*, *task*, and *event* are used interchangeably. In this paper, *activity* refers to the type of an *event*, while an *event* refers to a step in a workflow. Therefore, different events may have same activity type. We will avoid using *task* to eliminate the ambiguity.

Table 1 shows an example of an audit trail. In our work, we take into account three types of user operations: document editing, email sending, and web form submission. In the table, each row represents an audit trail entry. The first event in the audit trail says that Zhang sent an email to Johnson to inform him the acceptance of their paper. The second event says that Johnson was drafting a document to apply for a travel for a conference. The last event says that Bergman was booking an air ticket on an airline web site. The audit trail did not record explicit semantic labels for these entries, nor did it show which events are correlated for a workflow instance.

Table 1. An example of audit trail

Operation	**Time**	**Sender / Operator**	**Recipients**	**Content**
email	09/02/03 00:00:00	Zhang	M. Johnson	Hi Mike, our paper has been accepted …
doc	09/03/04 01:01:01	Johnson		The purpose of this travel is to learn the latest development in …
email	09/03/05 09:09:09	Johnson	S. Bergman	Hi Sarah, In August, I will have a trip to Boston for a conference …
web	09/03/06 09:09:09	Bergman		Air Canada ticket center…

Table 2 illustrates an example of an event log. In this table, we have four activity types *Apply*, *Approve*, *Decline,* and *Reimburse*, and two workflow instances: Instance 1 consists of three activities {*Apply*, *Approve*, *Reimburse*} and Instance 2 consists of two activities {*Apply*, *Decline*}. This table is a standard input for workflow mining algorithms that construct a graphic workflow model.

Table 2. An example of event log

Instance No.	Activity Types	Time
1	Apply	09/01/03 09:25:08
2	Apply	09/01/05 12:39:02
1	Approve	09/02/01 10:22:50
2	Decline	09/02/02 09:07:45
1	Reimburse	09/03/02 13:34:23

Problem statement Given an audit trail *AT*, convert it to an event log *EL*.

3 Converting Audit Trails to Event Logs

The mapping from an audit trail to an event log consists of two tasks: grouping related events for the same workflow instance and identifying the activity type for each event. Our integrated method includes the following steps: (1) Use a clustering method to identify the initial workflow instances. (2) Use a Naïve Bayesian classifier to identify the initial activity types. (3) Generate a *transition matrix* and *preceding matrix* based on the results from previous two steps. (4). Recluster the workflow instances with the transition matrix. (5) Reclassify activity types with the preceding matrix.

3.1 Discovering Workflow Instances

In this section, we address the problem of discovering process instances. A process instance corresponds to a single execution of a business process. Each process instance may consist of tens or hundreds of events, depending on the granularity level of the events. Unlike in the case of topic detection and tracking, where keywords are used to determine the similarity of two events [3], in workflow mining, we take the approach of viewing the similarity between events within a workflow instance being determined by the named entities contained in the artifacts of the events. For example, in the case of a travel application, every artifact (emails, documents, webpage forms) involved may include the name(s) of the applicant(s), the destination(s), and the dates for departure and return. Therefore, we use these named entities in the events to group events into instances. We treat the named entities as symbolic values; therefore the similarity of the named entities can be defined as $sim(e_1, e_2) = \frac{1}{K}\sum_{i=1}^{K}\frac{|E_{1i} \cap E_{2i}|}{|E_{1i} \cup E_{2i}|}$, where e_1 and e_2 are two events, K is the number of the entity types considered, E_{1i} and E_{2i} are the named entities for the two artifacts for the entity type i, which are sets of words. For example, suppose we have two documents, each of which contains two types of entities: locations and persons' names. The first document contains destinations {Paris, France, Toronto} and the persons' names {John, Smith, Mary, Bergman}. The second one contains destinations {Paris, Toronto} and persons' names {John, Mike, Mary}. The similarity between the two documents in terms of the named entities is $sim(d_1, d_2) = 1/2*(2/3+2/5) = 0.53$.

The similarity between an event *e* and an instance *ins*, which is a set of events, is defined as $sim(e, ins) = \frac{1}{K}\sum_{i=1}^{K}\frac{|E_e \cap (\bigcup_{e'\in ins} E_{e'})|}{|E_e \cup (\bigcup_{e'\in ins} E_{e'})|}$.

Here we did not try other specialized similarity measures because specifying the similarity measure too much to improve accuracy is not our purpose. Instead, we would like to see how integration of the instance discovery and activity discovery affects each other in a more general situation.

The clustering algorithm for discovering workflow instance is presented in Figure 1. *Ins* refers to the set of instances to be identified. *ins* refers to an instance which is composed of a set of events. The algorithm processes the events in the chronological order. It first finds the instances identified up to the present that is most similar to the current event. If the similarity value between the event and the most similar instance is greater than a threshold, it assigns the event to the instance. Otherwise a new instance is created with this event as the first event in the instance.

Greater similarity threshold values result in more workflow instances to be discovered, each of which contains fewer events, while smaller threshold values result in fewer instances, each of which has more events.

```
Function InstanceClustering
Rank the events in the ascending order of timestamp
$Ins$ = { }  // Initialize the set of instances
for each event $e$ do
    $sim = \max_{ins \in Ins}(sim(e, ins))$;
    if $sim > threshold$    //The instance for the event is identified
        $ins\_select = \arg\max_{ins \in Ins}(sim(e, ins))$
        $ins\_select = ins\_select \cup \{e\}$
    else  //A new instance for the event is created
        $ins\_new = \{e\}$
        $Ins = Ins \cup \{ins\_new\}$
    endif
```

Fig. 1. Clustering algorithm for workflow instance identification

3.2 Discovering the Activities

For a specific workflow, the number of the activity types is fixed. Classification algorithms can be used to train a classification model based on the keywords in the artifacts to classify the events into activity types. We used a Naïve Bayesian classifier to identify activities due to its efficiency and ease of incorporating new features.

According to the Naïve Bayesian classifier, given a set of keywords w_1, w_2, …w_k associated with an event e, the probability of e being an activity A can be defined as $P(A \mid w_1,...w_k) \propto P(A)\prod_{i=1}^{k} P(w_i \mid A)$.

In our implementation, Laplace smoothing is used to avoid zero values for $P(w_i \mid A)$.

We assign the event to the activity with the maximum posterior probability $A = \arg\max_i P(A_i \mid w_1,...w_k)$.

3.3 Constructing the Transition Matrix and Preceding Matrix

In some cases, the named entities in the audit trail may not be enough to discover the workflow instances. For example, suppose there are two instances intervening together. Instance one is about John's trip to London, Ontario and Instance two is about his trip to London, UK. If he were to write an email about an expense claim for his trip, but only included his name and London as named entities, it would be difficult to say which process instance this event belongs to. However, if the current stages in the process of the two instances are known, it may help make the decision. Suppose the current stage of process instance one is *applying for trip* and that of instance two is *booking hotel*. It might be safe to say that this new event should belong to instance two because it is more likely that the reimbursement is done after booking a hotel room and/or flight. Similarly, combining the keywords and the sequence information can also help classify activities.

After identifying the workflow instances and activities in the first round as described in Sections 3.1 and 3.2, we can generate the initial structured event log. Based on the initial event log, we can construct an n times n transition matrix, where n denotes the number of activity types. The transition matrix indicates the probability that each activity is followed by each other activity in a particular process instance. Specifically, the entry of row i and column j records the probability that activity i is followed by activity j, denoted as $Follow(a_i, a_j) = P(a_ia_j)/ P(a_i)$. We also construct an n times n preceding matrix, where each entry $Preceding(a_i, a_j) = P(a_ia_j)/ P(a_j)$ represents the probability that activity a_j is proceeded by a_i.

3.4 Using the Transition Matrix for Reclustering Workflow Instances

The reclustering algorithm is identical to the initial clustering algorithm as shown in Figure 1 except that we replace Line 5 with $sim = \max_{ins \in Ins} (Follow(a(last(ins)), a(e)) * sim(e, ins))$ and replace Line 7 with $ins\_select = \arg\max_{ins \in Ins} (Follow(a(last(ins)), a(e)) * sim(e, ins))$, where $a(e)$ denotes the mapping from event e to activity type, and $last(ins)$ denotes the last event that has been grouped in the current instance ins up to present. It should be noted that in the second clustering process, the optimal similarity threshold is different from that for the initial clustering due to the introduction of the factor *Follow*.

3.5 Reclassification for Activity Discovery

With the event log obtained from the initial clustering and classification, we can reclassify the events to new activity labels with the adjusted probability estimation $P(A \mid w_1,\dots w_k, A_P) \propto proceed\ \ (A_P, A) * P(A)\prod_{i=1}^{k} P(w_i \mid A)$, where $P(w_i \mid A)$ and $P(A)$ are the same items as in the first classification, $A_P = a(last(inst))$ denotes the activity type of the last event grouped in the instance *inst* up to present, and $preceding(A_P, A)$ refers to the probability that activity *A* is proceeded by activity *A_P*. It can be seen that in training the second classification model, we can use the items in the initial classification models and only need to obtain the preceding matrix from the initial event log, which makes the reclassification process very efficient. This reclassification for activity types combines both the keywords of the documents and the sequence patterns of the events. Here we assume the Markov property of the sequence, i.e., the current activity is only dependent on the preceding one.

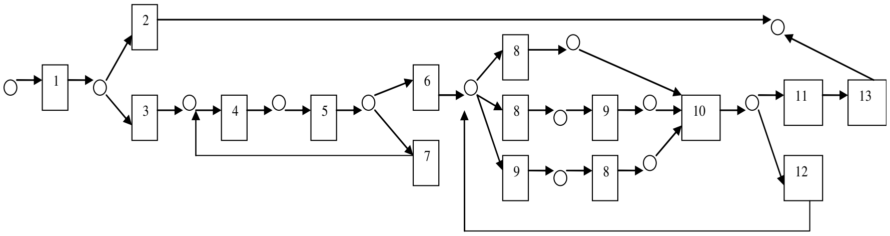
4 Experiments

We have conducted extensive experiments to evaluate the performance of our method. The experiments were implemented in Java and were performed on a Core 2 1.83GHz PC with 4GB memory, running on Windows XP.

4.1 Experiment Design

As pointed out by [7], the real email messages containing workflow is difficult to obtain due to privacy concerns, let alone the real data representing workflow which also contains other computer operations. Therefore we used simulated data for our experiments. First we investigated the common process of approving employee travel in an organization. We represented the simplified workflow model in a Petri Net as shown in Figure 2.

The simulated data sets were generated in two steps. First, structured event logs are generated from the workflow model. Then, operators, timestamps, named entities, and the keywords associated with each event were generated. Types of named entities we considered include traveler’s names, destinations, and dates of departure and return. Named entities and keywords can contain noise. We generated nine audit trail data sets with noise levels ranging between 10% and 90% with an increment of 10%. We consider three types of noise: insertion of a random words (or named entities) from a dictionary, deletion of keywords (or named entities), and replacement of keywords (or named entities) with other words in the dictionary (or other named entities of the same type). The three types of noise were added with the same probability. Each audit trail data set contains 100 instances with around 1400 events.



1. Employee applies for travel 2. Group leader denies application. 3. Group leaders approves application. 4. Travel authority books flight. 5. Travel authority informs employee on itinerary. 6. Employee approves itinerary. 7. Employee denies itinerary. 8 Secretary books hotel. 9. Secretary rents car. 10 Secretary informs employees the travel arrangement. 11. Employee approves travel arrangement. 12. Employee denies travel arrangement. 13. Employee claims travel expense

Fig. 2. Travel application workflow

We used the *F* measure to evaluate the instance clustering results. The *F* measure consists of two factors, *precision* and *recall*. In the scenario of clustering, *recall* represents how many object pairs that should be in same cluster are in the same cluster in the clustering results. The *precision* represents how many object pairs that are discovered in the same cluster are correct. The *F* measure is the harmonic mean of recall and precision.

We used *accuracy* to evaluate the classification of the activities. *Accuracy* is the ratio between the number of correctly classified objects to the number of all objects.

4.2 Experimental Results

Table 3 shows the *F* measure for the initial clustering results on the simulated data sets. Each row in the table represents a similarity threshold and each column represents a data set with different levels of noise. The best results for each data set are shown in bold face. Two observations can be obtained from the table. First, when we increase the noise level, the best derivable *F* measure decreases. This coincides with our intuition. Secondly, for the data set with higher level of noise, the best *F* measure values are obtained from smaller similarity thresholds. This is because when the noise level increases, the similarity values between events that belong to the same workflow instance decrease.

A second clustering is conducted on the best results of the initial clustering and the initial activity classification for each data set. Similarly, we vary the similarity threshold to obtain the best results for the second clustering. Figure 3 compares the results of initial clustering and second clustering. The X-axis denotes the level of noise for the data sets and the Y axis denotes the best *F* values obtained. If the activity labels are obtained from the initial activity classification, which is not perfect, the second clustering results are better when the noise level is below 30%. This means that if the quality of data is reasonably good, and accordingly the results of the initial clustering and classification are reasonably good, the second clustering will improve the results from the initial clustering. Otherwise, the second clustering will deteriorate the results. By intuition if the initial results are poor, and provide false information to the second clustering, it only makes things worse. We also compared these results with the second clustering when the activity labels are perfect. It can be seen that the second clustering results based on perfect activity labels are better than those of the initial clustering when the noise level is below 50%. Another observation is that second clustering with perfect activity labels almost always obtains better results than the second clustering with imperfect activity labels obtained from initial classification.

Table 3. Initial clustering results for instance identification

Noise T	10%	20%	30%	40%	50%	60%	70%	80%	90%
0.1	0.407	0.473	0.507	0.546	0.549	0.577	**0.600**	**0.513**	0.480
0.2	0.721	0.874	**0.942**	**0.907**	**0.746**	**0.579**	0.513	0.509	**0.514**
0.3	0.913	**0.910**	0.719	0.583	0.561	0.485	0.315	0.168	0.060
0.4	**0.970**	0.672	0.500	0.403	0.274	0.150	0.067	0.026	0.013
0.5	0.774	0.501	0.339	0.221	0.103	0.037	0.014	0.008	0.010
0.6	0.606	0.359	0.198	0.093	0.026	0.006	0.003	0.001	0.002
0.7	0.482	0.232	0.087	0.023	0.004	0.000	0.001	0.001	0.001
0.8	0.351	0.106	0.020	0.002	0.000	0.000	0.000	0.000	0.000
0.9	0.182	0.024	0.002	0.000	0.000	0.000	0.000	0.000	0.000

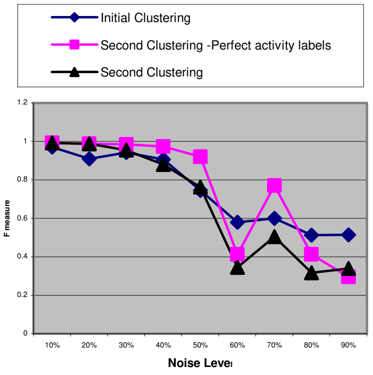


Fig. 3. Comparison of initial clustering and second clustering for instance identification

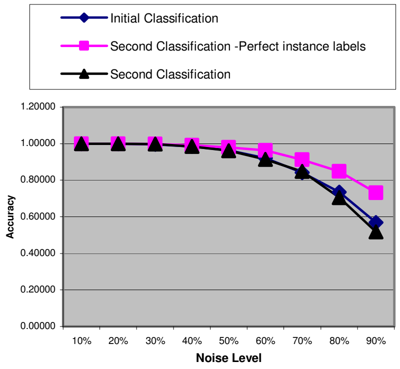


Fig. 4. Comparison of the initial and second classifications for the activity identification

Figure 4 presents the results for activity classification. It can be seen that when the noise level is below 70%, there is no significant difference between the initial classification and the second classification based on the imperfect instance labels obtained from the clustering process. When the noise level is above 70%, the second classification obtained worse accuracy than the initial classification. This is because the preceding activity identified is inaccurate such that it provides wrong information for the second classification. We also compared the initial classification and the second classification which is based on the perfect instance labels. It shows that the information for perfect labels for instances did improve the performance for the second classification in all noise levels. The experiments show that if the clustering method obtains instances with sufficiently good results, it improves the activity classification results. Otherwise, it could deteriorate the classification results.

5 Discussion and Conclusions

We worked on the problem of identifying instances and activities of workflows from unstructured data, and showed that integration of the two tasks has the potential to improve performance for each other, when they provide sufficiently accurate information to each other. Experimental results show that the integration of activity identification and instance identification is a double-edged sword. When the initial classification and clustering results are good enough, the second clustering and classification will obtain better results. Otherwise, performance deteriorates. Answering the question about how to define a “good enough” situation is our future work. Also we will apply this method to real data in the future.

References

[1] 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.

[2] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. *Proceedings of the 6th International Conference on Extending Database Technology*, 469-483, 1998.

[3] Topic Detection and Tracking: Event-Based Information Organization J Alan (Ed.), Kluwer Academic Publishers, 2002.

[4] V. R. de Carvalho and W. W. Cohen. On the collective classification of email "speech acts". *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2005), 345-352, 2005.

[5] J.E. Cook and A.L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.* 8(2): 147-176, 1999.

[6] G. Greco, A. Guzzo, G. Manco, D. Saccà. Mining frequent instances on workflows. *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 209-221, 2003.

[7] R. Khoussainov and N. Kushmerick. Email Task Management: An Iterative Relational Learning Approach. Second Conference on Email and Anti-Spam (CEAS), Stanford University, California, USA, 2005.

[8].N. Kushmerick and T.A. Lau. Automated email activity management: An unsupervised learning approach. Proceedings of the 2005 International Conference on Intelligent User Interfaces, 67-74, 2005.

[9] R. Silva, J. Zhang, and J.G. Shanahan. Probabilistic workflow mining. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 275-284, 2005.