Using Suffix-Tree to Identify Patterns and Cluster Traces from Event Log

Xiaodong Wang[2], Li Zhang[3], and Hongming Cai[1]

[1] School of Software, Shanghai JiaoTong University, Shanghai, China
[2] University of Mannheim, Mannheim, Germany
[3] IWW Institute, University Karlsruhe, Karlsruhe, Germany
wangxd.sjtu@googlemail.com, hmcai@sjtu.edu.cn, Li.Zhang@kit.edu

Abstract. Process mining refers to the extraction process models from event logs. Traditional process mining algorithms have problems dealing with event logs that are produced from unstructured real-life processes and generate spaghetti-like and incomprehensible process models. One means making traces more structural is to extract commonly used process model constructs (common patterns) in the event log and transform traces basing on such constructs. Another way of pre-processing traces is to categorize traces in event log into clusters such that process traces in each cluster can be adequately represented by a process model. Nevertheless, current approaches for trace clustering have many problems such as ignoring context process and huge computational overhead. In this paper, suffix-tree is firstly utilized for discovering common patterns. The traces in event log are transformed with common patterns. Thereafter suffix-trees are applied to categorize transformed traces. The trace clustering algorithm has a linear-time computational complexity. The process models mined from the clustered traces show a high degree of fitness and comprehensibility.

Keywords: Trace clustering, Suffix tree, Process mining.

1 Introduction

Today's information systems are logging events that are stored in so-called "event logs". For example, any user action is logged in ERP systems like SAP R/3, workflow management systems like Staffware, and case handling systems like Flower. An event log corresponds to a bag of process instances of a business process. A process instance is manifested as a trace which is an ordered list of activities. Process mining aims at a fine grained analysis of processes based on such event log [1]. It can deliver valuable and factual insights that show how processes are being executed in real life. Event logs are generally expected to be derived from well-structured processes. However, real-life business processes tend to be less structured than expected. Traditional process mining algorithms have problems with such unstructured processes and generate incomprehensible process models. In an event log, there can be instances where the system is subjected to similar execution patterns. Discovering of common patterns of invocation of activities in traces can promote comprehensibility of discovered process models. Another means to promote quality of process mining results is to categorize traces into clusters according to the prescribed characteristics of processes, so that complexities of traces in each cluster can be reduced and the resultant process models have more comprehensibility. In this paper a context aware approach to categorize traces into clusters is proposed. We first define patterns which commonly occur in traces. Suffix-tree is employed to discover common constructs (subsequences) in traces. Then the traces in event log are transformed as sequences of activities and patterns. A suffix-tree based approach is used to categorize transformed traces into clusters. This approach has a linear-time complexity and incorporates context information and execution order of processes during the trace clustering. We implemented the approach in the ProM framework[1] and evaluated the effectiveness of the approach through the goodness of mined process models.

2 Related Work

Greco et al. [2][3] proposed an approach to mine hierarchies of process models that collectively represent the process at different levels of granularity and abstraction. Jagadeesh et al. [4] proposed the definitions of context-aware patterns in traces and developed an iterative method of transformation of traces which can be applied as a pre-processing step for process mining techniques, yet the patterns were not fully considered for the clustering of transformed traces in this approach.Data clustering is one of the most important fields of data mining [5]. One of the most often used techniques for analyzing traces is to transform a trace into a vector, where each dimension of the vector corresponds to an activity. Song et al. [6][7] have proposed the idea of clustering traces by the combination of different perspectives of the traces as the feature vector. Though this combined approach might yield better results than before, such data modehas a few drawbacks. For example, the context information of process and execution order information are lost during the traces clustering. In [8][9], the generic edit distance based approach to trace clustering is proposed. However, the computational overhear of these approaches is still large.

3 Common Patterns in Traces

There are always special constructs in process models, such as loops and parallel constructs, etc. These constructs or abstract processes manifest themselves as different patterns in traces. Thereafter we need pre-process traces with such patterns, so as to improving the goodness of process mining results. In this section, we propose context-aware patterns of traces. The basic idea is to consider sub-sequences of activities that are conserved in and across traces, which signify some sets of common functionalities of process models. [10].

[1] ProM is an extensible framework that provides a comprehensive set of tools/plugins for the discovery and analysis of process models from event logs. See `http://www.processmining.org`

Let $\mathcal{A}$ denote set of activities.$\mathcal{A}$+is the set of all non-empty finite sequences of activities from $\mathcal{A}$. A trace, *T* is an element of $\mathcal{A}$. For $i \le j$,$T(i,j)$denotes the subsequence from thei^{th} position to the j^{th} position in the trace *T*. An event log$\mathcal{L}$, corresponds to a set of traces from$\mathcal{A}$+.

Simple loops manifest as the repeated occurrence of an activity or subsequence of activities in the traces. Thereafter, the tandem array of subsequence can be defined as:

*Definition 3.1 Tandem Array:*A tandem array in a trace *T* is a subsequence *T(i,j)*of the form α^k with$k \ge 2$where α is a sequence that is repeated *k* times. The subsequence α is denoted as a tandem repeat type.

Definition 3.2. Primitive Tandem Repeat (PTR): A tandem repeat is called a primitive tandem repeat if and only if$\alpha$is not a tandem array.

Definition 3.3. Maximal Pair: A maximal pair in a sequence*T*. (1) is a pair of identical sub-words, extending s_1 and s_2 on either side would destroy the equality of the two strings ; (2) there are no two neighbor letters which are same in such string.

Definition 3.4. Maximal Repeat: A maximal repeat is defined as subsequence αthat occurs in a maximal pair.

*Definition 3.5. Primitive Repeat (PR):*A primitive repeat is defined as a maximal repeat, which does not contain any other maximal repeat.

Considering an event log $\mathcal{L}$*={aabcdbbcda, dabcdabcbb, bbbcdbbbccaa, aaadabbccc, aaacdcdcbedbccbadbebdc}* over the alphabet $\mathcal{A}$*={a, b, c, d, e}*, For the traceT_5 the set of primitive repeat is *{a,c}*; the set of maximal repeat is*{bd, cb, db, dc, cdc}*. The set of primitive repeats in trace$T_5$is*{bd, cb,db, dc}*. Table 1 depicts the single repeats, non-single repeats and basic repeats in the entire event log$\mathcal{L}$.

Table 1. Primitive tandem repeat, maximal repeat and primitive repeat in event log

Primitive tandem repeat	Maximal repeat	Primitive repeat
{a, b,c,cd, dabc}	*{bcd, bd, cb,db,dc,cdc}*	*{bd, cb,db, dc}*

Process model contains special constructs, e.g. parallels, fork, join, etc. The execution order of activities in these constructs may vary from one process instance to another. Hence different patterns can share a common repeat alphabet. As to the above example, *[{a, b, d, g, h}]={abdgh, adgbh}*.

*Definition.3.6. Primitive Tandem Repeat Alphabet Set (PTAS):*The primitive tandem repeat alphabetset is the alphabet set that corresponds to primitive tandem repeat.

*Definition.3.7. Primitive Repeat Alphabet Set (PRAS):*The features of this set are derived from primitive repeat set*PR*. The primitive repeat alphabetset is the alphabet set that corresponds to primitive repeat.

With the aforementioned sets of patters*PTR, PR, PRAS*, each trace is transformed into feature sequences. The repeated activities subsequences in a trace are replaced by the feature alphabet sets which occur at the same place.

4 Clustering Traces with Suffix Tree

Suffix tree can be also applied to find feature repeats, since it allows a particularly fast implementation of many import string operation. Simple repeats that exist across the traces in the event log can be discovered by concatenating the traces in the event log with a special delimiter. In this paper, we focus on patterns that manifest as primitive tandem repeats or primitive repeats, as we use such patterns to transform traces so as to improve the quality of trace clustering.

Trace Transformation with Feature Sets: Any clustering technique relies on four concepts: data representation model, similarity measures, clustering model and clustering algorithm that generates the clusters using the data model and similarity measures [11]. In our approach suffix-tree model is employed as data model for trace clustering. As above mentioned, traces cannot be directly used to construct suffix-tree because of special constructs in processes. Here the traces are transformed based on the feature sets which are defined in section 3.2.

Definition.4.1. Featured Trace (ft): A featured trace is a trace from event log, whose repeats are replaced by corresponding pattern sets.

Concretely, tandem repeated subsequences are replaced by primitive tandem repeats and primitive tandem alphabet sets; other repeated subsequences are replaced by primitive repeats and primitive repeat alphabet set.

Clustering Traces with Suffix Tree: After transformation of traces, suffix tree is used to cluster featured traces. Zamir and Etzioni proposed Suffix Tree Clustering algorithm (STC) for Web-document clustering [12]. In our context, a phrase is an ordered sequence of one or more letters and feature alphabet sets. The logical steps of clustering traces analogically include identifying base clusters and combining base clusters into final clusters. A differentia between trace clustering and document clustering is that the number of activity types is finite. Therefore a trace is often made up of many repeated subsequences. For this reason such repetitious occurrence of subsequence should be considered during the clustering.*Identifying Base Clusters*: As presented in section 4.1, traces in event log are transformed into featured traces. The identification of base clusters can be viewed as the creation of an inverted index of phrases for featured trace collection. We treat featured traces as strings of letters and patters sets, thus suffixes contain one or more letters and feature alphabet sets. Figure 1 shows the suffix tree of the examples in section 3.

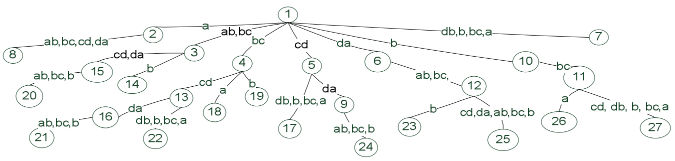

Fig. 1. Building base clusters through suffix tree

***Combining Base Clusters*:** After identifying base clusters, similar base clusters are combined. One subsequence may appear in more than one trace, and traces may share more than one common subsequence. To avoid the proliferation of nearly identical clusters, the high overlapped base clusters should be merged. Thereafter we use an equivalent of a single-link algorithm to group base clusters into end clusters. For the above example, the end clusters are *{{1,2},{1,3},{2,3}}*.

5 Evaluating the Goodness of Clusters and Experiments

In this section we introduce the criteria to evaluate the goodness of clusters in our approach and corresponding experiments result. To evaluate the significance of the clusters formed, one can compare the process models that are discovered from the traces within each cluster. Good clusters tend tocluster trace such that :(1)The process models mined show a high degree of fitness. (2)The process models mined are less complex [8]. To evaluate the result of trace clustering, the approach was implemented as a plug-in in the framework of ProM. An event log of telephone repair is employed as data set. This event contains a total of 1104 process instances. Random subsets of this data set are chosen for analysis. For each sub set of instances, three clustering techniques are applied for clustering (1) k-gram model approach; (2) Generic edit distance approach; (3) Suffix-tree approach. Process models of clusters were generated through the Alpha++ mining algorithm [13]. The fitness of process model was calculated in the Conformance Checker plug-in in ProM. Petri-Net Complexity Analysis plug-in was used to generate metrics of control flows, such as join, split, arcs in process models. Figure 2a depicts weighted average fitness of the process models mined from the trace clusters. Figure 2b shows the execution time of clustering with three techniques. It shows that our approach has approximate linear computational complexity.

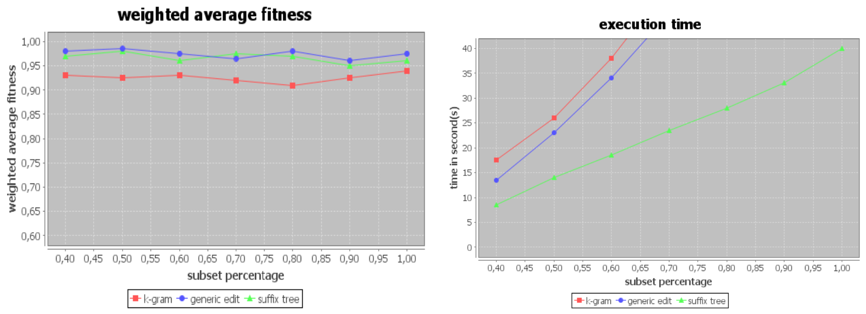

Fig. 2. Weighted average fitness of three trace clustering techniques and execution time of experiments

6 Conclusions and Future Directions

In this paper, a suffix tree based approach for trace clustering is proposed. Patterns of subsequence in traces are introduced to aid to representing traces in clustering. Based on pattern sets, traces are transformed so as to be clustered with suffix tree. It was shown that the proposed approach has good clustering results and is faster than other trace clustering algorithms. Identifying such activities' relationship can improve the efficiency of process mining and help discovery more semantic information of process models.

Acknowledgment. This research is supported by the National Natural Science Foundation of China under No.70871078, 71171132, the National High Technology Research and Development Program of China ("863" Program) under No.2008AA04Z126, and Shanghai Science and Technology Projects 09DZ1121500.

References

1. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
2. Greco, G., Guzzo, A., Pontieri, L.: Mining Hierarchies of Models: From Abstract Views to Concrete Specifications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 32–47. Springer, Heidelberg (2005)
3. Greco, G., Guzzo, A., Pontieri, L.: Mining Taxonomies of Process Models. Data Knowl. Eng. 67(1), 74 (2008)
4. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009)
5. Jain, A.K., Murty, M.N., Flynn: Data Clustering: A Review. ACM Computing Surveys 31(3), 264–323 (1999)
6. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)
7. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Disco-covering Expressive Process Models by Clusering Log Traces. IEEE Trans. Knowl. Data Eng., 1010–1027 (2006)
8. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Context Aware Trace Clustering: Towards Improving Process Mining Results. In: Proceedings of the SIAM International Conference on Data Mining, SDM, pp. 401–412 (2009)
9. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)
10. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 170–181. Springer, Heidelberg (2010)
11. Hammouda, K.M., Kamel, M.S.: Efficient phrase-based document indexing for web document clustering. IEEE Transactions on Knowledge and Data Engineering 16(10), 1279–1296 (2004)
12. Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 46–54 (1998)
13. Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining Process Models with Non-Free Choice Constructs. Data Min. Knowl. Discov. 15(2), 145–182 (2007)