Efficient Searching of Action Traces Based on MD5

Juntao GAO, Hongbo ZHOU

School of Computer and Information Technology, Northeast Petroleum University, Daqing, 163318 China

Summary
Searching of action traces is one of basic operators which is useful in several scenarios during business process management, such as process mining, process model search, process re-engineering and so on. One of the challenges to search action traces is that the scope of searching is too wide to handle and the time consuming is beyond tolerance. Firstly, a framework of methodology to search action traces is proposed; secondly, the algorithm of MD5 are employed to construct index; thirdly, the algorithm to compute similarity between action traces are discussed ; Finally, an experiment is given to evaluate the methodology.

Key words:
action traces, firing sequence, index.

1. Introduction

Through the application of Business Process Management(BPM), kinds of process-aware information systems, such as ERP (Enterprise Resource Planning), SCM (Supply Chain Management), PDM (Product Data Management) are employed, then large amount of action traces have been accumulated in various information systems. An action trace, also called firing sequence in the domain of Petri nets, is a finite or infinite sequence of activities that denotes the order in which the execution of activities starts in an instance of the process.

These action traces are important intellectual assets of organizations, so a deep insight into these action traces and their mutual relationship is necessary to business process management activities. There are various applications in business process management that require measuring the similarity between action traces, such as process mining, process model search, process reengineering and so on. For example, the sets of action traces of business processes are compared to calculate compliance and maturity of an actual process model to a reference model in process reengineering [1]. In this context, the compliance degree and the maturity degree of two traces are defined based on their longest common subsequence. After that, the overall compliance and maturity degree between two models are calculated by summing up the maximum compliance and maturity degree of traces belong to them. Another example is to align action traces in quantitative analysis method of business process [2]. The actual action traces are compared to the simulated traces from predefined business process models. According to the result of action trace comparing, the bottleneck and critical path are identified. In paper [3], the set of traces are used to construct a reference similarity of business processes.

A key operation required by nearly all these techniques is the searching from action traces repository. Given an action trace (the trace query), we are concerned with finding all action traces in the repository that contain this fragment. Because of the number of action traces is usually very huge, MD5 algorithm is employed to efficiently search action traces.

This paper is constructed as follows: In the next section, the framework of methodology to search action traces is proposed in section 2. Section 3 discussed the method to construct index of action traces repository. Section 4 presents the algorithm to compute similarity between action traces. At last, the conclusion is drawn.

2 The Framework of Methodology

In this paper, focus is on the provision of efficient support for querying action trace repositories. Given an action trace (the trace query), we are concerned with finding all action traces in the repository that contain this fragment. The complexity of finding all action traces is known to be high of time complexity. To overcome this issue, we propose a two phase methodology that reduces the number of action traces needed to be checked for similarity.

As illustrated in Fig 1, firstly, we filter the trace repository through the use of indexes and obtain a set of candidate action traces. The query of action traces is denoted as Q in Fig 1. MD5 algorithm is employed to compress the query trace into a code. The index is constructed as a B+ tree in which the MD5 code is selected as key terms.The result of first phase in this methodology is a subset of action trace repository.

Secondly, we apply a similarity measure algorithm to compare those action traces in the result of the first phase. The advantage of using indexes is that the similarity measurement is only performed on a subset of the models in the repository, which is usually much smaller than the total number of action traces in the repository. The matrix of action trace is constructed and the reference similarity measurement is given in the following sections.

Manuscript received September 5, 2015
Manuscript revised September 20, 2015

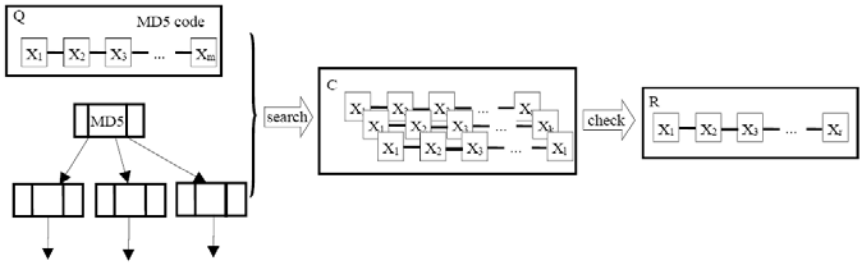

Fig. 1 the framework of methodology

In this paper, action traces are defined as sequences of actions. Suppose Σ denotes the universe of actions, the action trace α is defined as

$\alpha =< x_1, x_2, ..., x_m >$ where $x_i \in \Sigma$, $i \in [1, m]$

x_i denotes the i-th action in trace α, i=1,2,...,n. $|\alpha|$ denotes the length of trace α, which is the number of actions in α.

In the second phase, the similarity between action traces is computing according to the semantics of actions and the sequence among them.

Suppose Σ denotes the universe of actions, there exist two traces α and β,

$\alpha =< x_1, x_2, ..x_m >$ and $x_i \in \Sigma, i \in [1, m]$

$\beta =< y_1, y_2, ..., y_n >$ and $y_j \in \Sigma, j \in [1, n]$

x_i denotes the i-th action in trace α, i=1,2,...,n. $|\alpha|$ denotes the length of trace α, which is the number of actions in α. y_j denotes the j-th action in trace β, j=1,2,...,n. $|\beta|$ denotes the length of trace β, which is the number of actions in β.

The commonality of α and β is depicted by $common(\alpha, \beta)$,

$common(\alpha, \beta) =< X \bigcap Y, lct >$, X is the action set of α, Y is the action set of β. Because the action may occur more than once, X and Y are both multisets. The commonality of α and β includes two parts: one is the common action set $X \bigcap Y$, the other is the longest common subtrace, lcs for short, from the common action set. It is deployed to measure the similarity of the order of action occurring.

The combination of trace α and β is depicted by $description(\alpha, \beta)$,

$description(\alpha, \beta) =< X \bigcup Y, \{\alpha, \beta\} >$,

The combination of α and β also includes two parts, one is the union of action set $X \bigcup Y$, the other is two alternative action sequences $\{\alpha, \beta\}$.

According to information theory [5], the reference similarity of action traces is:

$$sim(\alpha, \beta) = \frac{\log P(common(\alpha, \beta))}{\log P(descritipn(\alpha, \beta))} \quad (1)$$

If the probability of trace is known, the above formula can be computed using the following formula.

$$sim(\alpha, \beta) = \sqrt{\varepsilon \times \left(\frac{|X \bigcap Y|}{|X \bigcup Y|}\right)^2 + \varphi \times \left(\frac{|lct|}{|X \bigcap Y|}\right)^2} \quad (2)$$

where, $\varepsilon \geq 0, \varphi \geq 0$ and $\varepsilon + \varphi = 1$.

The value of ε and φ is determined by the amount of information contained in the action sets and their orders. Generally, the cardinality of universal action set is very large, so the probability of common actions occurring is very little and the amount of information contained in action sets is very large. While given the common action set, the probability of the same order occur is relatively big and so the amount of information contained in it is relatively less. Therefore, ε is bigger than φ. The process of compare commonality is discussed in section 4.

3 Index Construction

To enhance the efficiency of action trace indexes, the items are not stored directly. B+ trees are employed to store items[4]. A B+ tree is an n-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children. Each node is denoted as a tuple $< IN, AT >$, in which IN is the MD5 code and AT is the action trace.

As we known, the MD5 message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format

as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity. In this methodology, MD5 message-digest algorithm is employed to compress the action trace to a 32 digit hexadecimal number[5].

In our methodology, the roots of B+ trees are kept in memory while the other nodes are stored on disks. Cache memory can be used for B+ tree nodes and inverted lists to further improve the efficiency. As we aim to minimise retrieval time, it is beneficial to keep the depth of B+ trees minimal and to avoid hash collisions as much as possible. As labels of actions may be arbitrarily long strings, a MD5 function is applied to map such strings to numbers in order to save space and speed up querying.

4 Definition of Similarity

Similar traces are defined based on the action similar. In reality, the action trace is identified by short text. The result of comparison between action x_i and y_j is not a binary value. Therefore, the traditional method to compute the union and intersection of action sets does not work in this case. In this section, an alignment-based method is discussed to compute the commonality of action set. This method involves three steps: (1) construct the similarity matrix; (2) pick up best matching; (3) computing the commonality. Next, each step is going to be explained.

4.1 Constructing the Similarity Matrix

Because the identifiers of action may be made by different systems, the vocabulary employed to identify actions may well be different. Therefore, the synonyms and homonyms are inevitable and make it difficult to compare the actions. In order to address the semantic heterogeneity, edit distance [6] and WordNet [7] is combined to measure the initial action similarities, which is the seeds of similarity matrix to start the iteration. Next, the process iteration is presented.

Given two actions $x \in \alpha$ and $y \in \beta$, the semantic similarity is defined as $SimA(x, y) \in [0,1]$. It is a total function over $\alpha \times \beta$ and determined by an iterative computation to simulate the flooding phenomenon of similarity among action traces. The flooding phenomenon means that if x_i is much similar to y_j then the similarity between x_{i-1} and y_{j-1} raise, as well as the similarity between x_{i+1} and y_{j+1}

$$SimA_k(x_i, y_j) = \omega SimA_{k-1}(x_i, y_j) + \varphi SimA_{k-1}(x_{i-1}, y_{j-1}) + \lambda SimA_{k-1}(x_{i+1}, y_{j+1}) \quad (3)$$

If $i = 0$ or $j = 0$, there is no pre-action of action x_i or no pre-action of action y_j. Therefore,

$$SimA_k(x_i, y_j) = \omega SimA_{k-1}(x_i, y_j) + \lambda SimA_{k-1}(x_{i+1}, y_{j+1}) \quad (4)$$

If $i = m$ or $j = n$, there is no sub-action of action x_i or no sub-action of action y_j. Therefore,

$$SimA_k(x_i, y_j) = \omega SimA_{k-1}(x_i, y_j) + \varphi SimA_{k-1}(x_{i-1}, y_{j-1}) \quad (5)$$

The similarity of action pair (x_i, y_j) is determined by the last result of iterative computation, its pre-action and sub-action. Fox example, if the names of two actions are different, but their pre-action and sub-action are same, their behavior must be more similar than their names. On the contrary, if their name seems alike, but pre-action and sub-action are absolutely different, their behavior must be less similar than their names. The first action in a trace has no pre-action and the last action in a trace has no sub-action. Therefore, the algorithm to compute these two kinds of action similarity is different from the ordinary actions. α, φ, λ are called propagation coefficients ranging from 0 to 1. They can be computed in many different ways.

After once flooding computation, the sum of similarity may shift a little. In order to keep the invariance of the sum of similarity, the result should be normalized, using the following formula.

$$SimA_k(x_i, y_j) = SimA_k(x_i, y_j) \times \frac{\sum_{i\in[1,m],\, j\in[1,n]} SimA_k(x_i, y_j)}{\sum_{i\in[1,m],\, j\in[1,n]} SimA_{k-1}(x_i, y_j)} \quad (6)$$

The computation is performed iteratively until the Euclidean length of the residual vector $\Delta(Simx_n, Simy_{n-1})$ becomes less than ε for some $n > 0$. If the computation does not converge, it is terminated after a certain number of iterations. The final similarity of actions is denoted as $SimA(x_i, y_j)$.

4.2 Picking up the best Matching

In the last section, all action pairs are assigned values to denote similarities. This section focuses on the issue how to pick out the best matching M, which maximizes the sum of similarity degrees. A mapping is a subset of activity pairs (x_i, y_j), in which x_i is from trace X and y_j is from trace Y. The combinatorial explosion of the number of mappings makes the issue difficult to resolve. Therefore, Hungarian algorithm [8] is expanded to solve the problem Here, the validity of the algorithm is discussed.

(1) Constructing similarity matrix. Computing the similarity of the action x_i in trace α and the action y_j in trace β .Then assigning the value to the element (i, j) in similarity matrix.
(2) Subtracting off the row min from each row.
(3) Subtracting off the column min from each column.
(4) Starting with the row or column with the least number of zeros, marks one certain zero element and redlines the row and the column where the marked zero element exists.
(5) Repeating step 4 until each zero element is marked or redlined. If the number of marked zero elements is $min(m, n)$, match the action of trace α in the row in which the zero element exists to the action of trace β in the column in which the zero element exists. Otherwise, go to step 6.
(6) Mark all rows without marked zero with *, and then mark all zero elements in rows with *, and mark all zero elements in columns with mark *, until mark * can not be added.
(7) Redline all the rows and columns without *.
(8) Identify the least one among the elements uncovered by lines and denoted by x_{ij} .
(9) Subtract x_{ij} from the rows marked with *, and subtract x_{ij} from the rows marked with *, return to step 4.

4.3 Measuring the Similarity of Action Traces

Using dynamic programming technology, such as Needleman-Wunsch Algorithm [9] and Smith–Waterman algorithm [10] the longest common subtrace is determined. The length of longest common sub-traces between action traces α and β is denoted as $lct(\alpha, \beta)$. If there is no common sub-traces between α and β, $lct(\alpha, \beta) = 0$. If α is same as β, $lct(\alpha, \beta) = len(\alpha) = len(\beta)$,where $len(\alpha)$ denotes the length of action trace α.
According to the best matching M , the scores for aligned actions are computed as following formula.

$$lct(x_i, y_j) = \begin{cases} lct(x_{i-1}, y_{j-1}) + 1 & (x_i, y_j) \in M \\ Max(lct(x_{i-1}, y_{j-1}), & \\ lct(x_{i-1}, y_j), lct(x_i, y_{j-1})) & (x_i, y_j) \notin M \end{cases} \quad (8)$$

Then the length of longest common sub-traces between action traces α and β can be computed using classical Needleman-Wunsch Algorithm.
Adopting the result of measure commonality depicted above, the similarity of action traces drills down to the following algorithm.

$\alpha =< x_1, x_2, .. x_m >$ and $x_i \in \sum, i \in [1, m]$

$\beta =< y_1, y_2, ..., y_n >$ and $y_j \in \sum, j \in [1, n]$

$$sim(\alpha, \beta) = \sqrt{\varepsilon \times SimASet(\alpha, \beta)^2 + \phi \times \frac{|lct|^2}{|\alpha \bigcap \beta|}} \quad (9)$$

5 Conclusion

In this paper a new approach is proposed to search action traces. Not only the sequence similarity but also the index construction is considered in this paper. The approach is more adaptive to the real application scenarios, in which the action is described by a textual message. So far, the work in this paper has been applies into a project of cross organization ERP implementation. In the future, the method still needs more projects to verify.

Acknowledgments

The research is supported by the Education Department of Heilongjiang province science and technology research projects (No. 12541094).
The research is also supported by the Northeast Petroleum University youth science and technology research projects (No. 2013NQ118).

Reference

[1] Gerke, K., Cardoso, J.,Claus, A.: Measuring the compliance of processes with reference models. In: On the Move to Meaningful Internet Systems – Confederated International Conferences 2009, Proceedings, Part I, Springer(2009) :76-93

[2] LI Yan, FENG Yu-qiang. A Quantitative Analysis Method of Business Process based on Sequence Alignment. System Engineering Theory and Practice,2007, 27(4):54-61

[3] Haiping Zha, Jianmin Wang, Lijie Wen, Chaokun Wang, Jiaguang Sun . A workflow net similarity measure based on transition adjacency relations. Computers in Industry.2010, 61 (5) :463-471

[4] Navathe, Ramez Elmasri, Shamkant B. (2010). Fundamentals of database systems (6th ed.). Upper Saddle River, N.J.: Pearson Education. pp. 652–660.

[5] Ciampa, Mark (2009). CompTIA Security+ 2008 in depth. Australia ; United States: Course Technology/Cengage Learning. p. 290.

[6] P. Bouguet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krotzscn, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris, "Specification of A common framework for characterizing alignment," Knowledge Web Consortium 2005.

[7] P. Pantel and D. Lin, "Discovering Word Senses from Text," presented at Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2002.

[8] Harold W. Kuhn. "The Hungarian Method for the assignment problem,"Naval Research Logistics Quarterly, 2: 83-97, 1995

[9] Bergroth, L., Hakonen, H., Raita, T.: A survey of longest common subsequence algorithm. String Processing and Information Retrieval, International Symposiumon (2000). 39-48

[10] Smith, Temple F.; and Waterman, Michael S.. "Identification of Common Molecular Subsequences". Journal of Molecular Biology(1981) 147: 195–197.

Juntao GAO received the PhD. degrees in Computer Science from BeiHang University in 2009. During 2009-2014, he stayed in Northeast Petroleum University to teach software engineering. His interest and research areas include process modeling, software requirements, semantic computing. Email: gjt@nepu.edu.cn.

Hongbo ZHOU master, he is now works in the Northeast Petroleum University. His interest and research areas include information retrieval, clustering, and data integration. Email:jiessie9@126.com