Web Service Usage Mining: Mining For Executable Sequences

MOHSEN JAFARI ASBAGH[†§], HASSAN ABOLHASSANI[§]

[†] ISIRAN
Center of Research and Development
in Basic Systems
Tehran
IRAN

[§] Computer Engineering Department
Sharif University of Technology
Azadi Ave., Tehran
IRAN

Abstract: As service world becomes bigger, behavior of people in this world becomes interesting and analysis of usage sequences can yield useful information about web services and the way they are used. Application of data mining and web mining techniques in the field of web services is introduced in order to discover interesting patterns among web services usage and interactions. In this paper, we introduce the concept of the executable sequence of the web service operations and propose an efficient algorithm for mining sequential patterns among these sequences which shows good performance regarding basic sequential pattern discovery algorithm.

Key-Words: Data Mining, Sequential Pattern Analysis, Web Service, Web Service Usage Mining, Executable Sequence, Frequent Pattern Discovery.

1 Introduction

Using web services in various business areas such as e-commerce and e-business is getting more popular and services provided by different providers can be used by intended users all around the world. As the ability of the web service technology in the fulfillment of the needs of the IT world becomes increased, more users get encouraged to import this approach in their personal activities such as hotel reservation and their technological activities such as business automation. This increase in the use of the web services, forces web service providers to enter in a competitive race in producing and providing more and more services in order to achieve success and gain potential benefits. In this way, every approach which help service providers to better understand market trends and help users to better know what is accessible to use, is considered as a source of power for users as well as providers and consequently helps the web service technology to reach success.

One approach that can be helpful in this way is to apply data mining and web mining techniques in various fields of web service area which provides ability to analyze patterns and behaviors in a web service community. The application of various data mining techniques in the field of the web service is investigated in [1]. Q. A. Liang et al. named application of data mining techniques in the field of the services as Service Mining [2].

Sequential pattern analysis initiated by Agrawal in [3] is one of the data mining techniques which mines frequent sequential patterns in time series database. In the web service literature, sequential pattern analysis can be applied in order to find sequences of web service operations which are called by web service users. This yields in hand knowledge about the patterns of interests among service users and indicates that which services have high correlation with each other. This knowledge can provide decision makers with useful information which in turn can guide them to make important decisions.

Sequential pattern mining among web service operations is one of the tasks of the *web service usage mining* –borrowed from web usage mining in the field of web mining [4]. Searching for these sequential patterns is done through logs made by service hosts from the usage of the web services. There is not such a log yet, but we hope that it will be provided in a close future.

In this paper, we suggest *AprioriAll* algorithm [3]-[5] as a basic algorithm for mining frequent sequences of web service operations. Also the appropriate log format on which this algorithm can work is proposed. We also introduce the concept of *executable sequence* of web service operations as a special web service usage sequence in which the succedent operation in the sequence uses output of the precedent operation as one of its inputs. In order to mine sequential patterns in executable sequences, we make an improvement on the basic algorithm via some optimizations that can help to get better timing in the execution of the mining algorithm.

The remainder of this paper is structured as follows. Section 2 addresses the motivation against executable sequence pattern analysis. In section 3, we first propose a log format for recording the usage of the web service operations, and then we introduce a basic algorithm for mining sequential pattern in the web service log. Section 4 contains our improved algorithm for mining sequential patterns among executable sequences. Section 5 presents some experiments and evaluation of our algorithm. Some related works are appeared in section 6 and section 7 states the conclusions and future works.

2 Motivation

Automated web service composition has taken much effort in recent years [6]-[7]. Q. A. Liang et al. propose a way toward web service composition based on users query via solving constraint satisfaction problem [2]. In their solution, service operations are represented in a template in which each node is a class of operations which have the same signature. In each composition candidate, one operation from each node can be selected. After that, candidate sequences of the operations are investigated in order to detect potential conflicts on their constraints. If such conflicts exist, the sequence is removed from the candidates.

Even though we succeed in discarding inconsistent sequence of the operations, what we should do if those remained possible choices be more than one? The authors don't address how to treat with this condition.

We believe that one approach in this situation is that a user based on the information which is provided by the service vendor, makes a choice best matching his/her needs. Feedbacks from these decisions can be used in order to provide prospective users with recommendations to make appropriate choices.

Here is where our executable sequential pattern mining can be helpful. Every possible choice of the user is an executable sequence. If our proposed executable sequential pattern mining algorithm be applied to the log of the previous executions, it can mine frequent executable sequences. Then, the possible choices can be ranked based on these frequent patterns and can be recommended to the user.

3 Web Service Usage Mining via Sequential Pattern Analysis

Web service usage mining is the discovery of interesting patterns among web service usage log. Usage patterns are the styles of the usage which occur frequently and indicate specific patterns of interest and high degree of co-occurrence between web services or web service operations. We put our focus on correlation between web service operations and we try to discover frequent sequence of web service operation calls.

Before any algorithm is developed in this context, an appropriate usage log must be at hand. Since such a log does not exist yet, first we introduce a format for the entries of the usage log of the web services that gives us sufficient information in our way to analyze patterns. Then, the *AprioriAll* algorithm [3]-[5] is rewritten with some minor changes to give it the ability of the operation over web service usage log.

3.1 Usage Log of Web Services

Since we are looking for the sequences which frequently occur in web service usage sessions (e.g. the interval starting at the time of the connection of a user or composition engine to web service host until the disconnection from it), sufficient information must be embedded in log entries in order to identify the user sessions. In addition, each log entry has to contain the information about the operation within the target web service which is called. We put all of these together and propose a structure for the log entries.

Each session starts and ends with *begin* and *end* entries respectively. When a session begins, an identifier –*sessionID*- is assigned to it which refers to that session and all the entries contained in it. In addition to these data items, the user of the session (session initiator) and the time of the session entries are also included. The proposed format for the begin and end entries is as follows:

[begin | end] – sessionID – user – timestamp

During the life time of a session, each entry includes the *sessionID* to identify the session to which the entry belongs. Two other data items that are contained in each log entry are the web service that is used -*service*- and the operation called within it (*operation*). A timestamp is also included in each entry. The structure of this entry is as follows:

sessionID – service – operation – timestamp

3.2 Sequential Pattern Mining within Service Usage

The *AprioriAll* algorithm [3]-[5] which has been developed for mining the sequential patterns among transactional data can be used as a basic algorithm for analyzing the web service usage and discovering the interesting usage patterns. The rewritten algorithm has been shown in Fig.1.

```
Algorithm WUM-SequentialPatterns
Input:
U: Web service usage log
min_support: Minimum support
Output:
Set of frequent sequences
Steps:
Begin
  L1 = The set of the frequent 1-itemsets;
  For (k=2; Lk-1 <> null; k++) do
  Begin
    Ck = Apriori-gen(Lk-1);
    For each candidate c in Ck do
      If support of c >= min _support then
        Add c to Lk;
  End
  Find maximal reference sequences from L;
End
Procedure Apriori-gen(Lk-1)
Begin
  Ck=null;
  For each operation set Li in Lk-1
    For each operation set Lj in Lk-1
    Begin
      If Li and Lj can be joined then
        Begin
          c1 and c2 = Li Join Lj;
          If c1 has no infrequent sub-sequence then
            Add c1 to Ck;
          If c2 has no infrequent sub-sequence then
            Add c2 to Ck;
        End
    End
  Return Ck;
End
```

Fig.1. Basic sequential pattern analysis algorithm among web service usage log.

A sequential pattern in a web service usage log denotes an ordered list of the web service operations that its occurrence count exceeds a specific threshold. In other words, its support must be higher than a threshold. This threshold is the input of this algorithm.

Apriori-gen procedure generates k-itemset (sequence of k operations) candidates by means of joining frequent sequences with length k-1. Those two sequences with length k-1 can be joined that either has just one different operation in the same position or the first k-2 items of one sequence be exactly the same as the last k-2 items of the other one. In case of the former condition, join operation returns two operation sets. This is a result of the fact that two different operations in the sets to be joined can be combined in two different orders and both of these orders must be added to candidate set.

4 Executable Sequential Pattern Analysis

In the data mining area, when talking about the data, we mean a huge collection of the naive data that contains millions of data records. Any process or lookup in such a huge data collection is a time and resource consuming task. Considering this situation, it's straightforward to guess that the major effort taken in the mining for sequential patterns belongs to searching through the data set in order to determine frequent k-itemsets among the candidate item sets. As stated, since searching in the large collection of the data is a time consuming task, the larger the size of the candidate set, the longer time the sequential pattern mining algorithm takes. Therefore, one way to improve the performance of such algorithms is to make reduction in the size of the candidate set.

When looking for the frequent *executable sequences*, we can detect that some sequences can't be frequent without searching in the data collection. We earlier introduced the concept of the executable sequence for web service operations as a sequence of web service operations that the output of the prior operation is consumed by the subsequent operation. Therefore the output of the prior service operation and one of the inputs of the subsequent one should be compatible. When there is no such compatibility between two web service operations in a specific order, we'll be sure that those two operations will not appear in that order within a session.

Using this heuristic, we can modify *Apriori-gen* procedure in such a way that when trying to join two operation sets, it checks whether the result of the join is an executable sequence or not. If the result is positive, it is added to candidate sets and otherwise it is discarded. This modification results in reduction in the size of the candidate set and yields improvement in the execution time of the mining algorithm. We call such a join operation as *enhanced-join* operation.

We developed the executable sequential patterns mining algorithm based on the idea mentioned above which its *Improved-Apriori-gen* procedure has been shown in Fig.2. The other parts of the algorithm are the same as the presented algorithm in Fig.1. Within *Improved-Apriori-gen* procedure, it is possible that one or both of the sets returned by *enhanced-join* operation be empty which indicates that the corresponding sequence was not an executable sequence.

```
Procedure Improved_Apriori-gen(L_{k-1})
Begin
  C_k=null;
  For each operation set L_i in L_{k-1}
    For each operation set L_j in L_{k-1}
    Begin
      If L_i and L_j can be joined then
        Begin
          c_1 and c_2 = L_i Enhanced-Join L_j;
          If c_1 is not empty
           & c_1 has no infrequent sub-sequence then
             Add c_1 to C_k;
          If c_2 is not empty
           & c_2 has no infrequent sub-sequence then
             Add c_2 to C_k;
        End
    End
  Return C_k;
End
```

Fig.2. Executable sequential pattern analysis algorithm among web service usage log.

5 Experiments and Evaluation

In order to evaluate our idea and to measure the gain in time yielded by our proposed improvement, due to the lack of the service usage log, we decided to do some experiments over an analytical log. To do this, we developed a program which randomly generates sessions containing the sequences with two, three, or four operation calls. The number of these sessions equals to 200,000 which results in approximately 500,000 log entries excluding begin and end entries. We establish a ratio between the executable sequences and the non-executable ones every time a log is generated. In order to make some sequences to satisfy the minimum support count, some of the candidate sequences are selected and randomly written over some occurrences of the other sequences. This makes some sequences to be more frequent than others and consequently sequential pattern analysis program can be done providing some results.

We select the execution time as the main criterion to be measured in order to evaluate our algorithm. This criterion also gives implicit information about the number of the candidate sequences examined during the execution of the algorithm which can be considered as another criterion. This is because of the fact that the log is traversed for each candidate sequence and other processing times are negligible regarding this traverse time. To better analyze the performance of our algorithm, we measure the execution time regarding two factors: the fraction of the number of the executable sequences over all the log sequences which we call log ratio and the minimum supports. We repeat the experiment for the nine different values of the log ratio starting from 0.1 up to 0.9 incremented by 0.1 in each step. For each value of the log ratio we must generate a new log. Furthermore, in each step we take five runs according to five different values for the minimum support: 0.04, 0.06, 0.08, 0.10, and 0.12. Therefore, the ongoing results arise from 45 different runs of the both of the basic and improved algorithm.

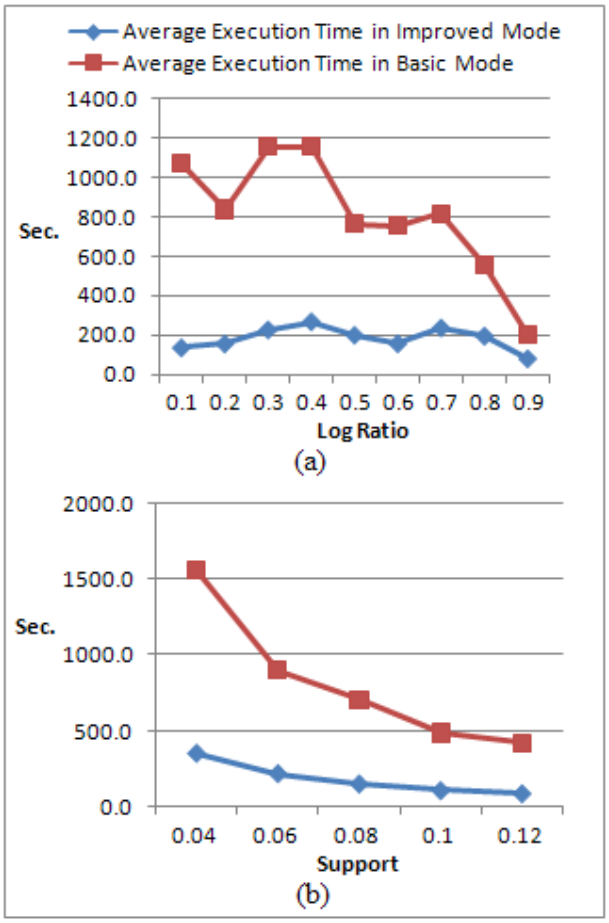

Fig.3. Execution time in seconds. (a) based on different log ratios. (b) based on different minimum supports.

Fig.3 illustrates the execution time of the algorithms in seconds. Fig.3 (a) compares two algorithms in the terms of the duration in seconds for different values of the log ratio and Fig.3 (b) represents the same comparison but for different support values. We expect that with the increase in the log ratio as well as the minimum support, the difference between the execution time of the two algorithms become decreased. This occurs because with the increase in the log ratio, the number of non-executable candidate sequences which impose useless effort in basic algorithm becomes reduced. In the other side, the higher value of the

minimum support reduces the number of both of executable and non-executable candidate sequences and consequently reduces the execution time of both of them as well as their difference.

This difference in the execution time gives a little information about the behavior of these two algorithms since although the difference decreases, but the execution time itself becomes decreased and this fact prevents us from doing an accurate inference. To achieve better evaluation, we measure the performance of our algorithm by means of the fraction of the time taken by basic algorithm which our improved algorithm takes. We simply divide the execution time of the improved algorithm by the execution time of the basic algorithm. We call this ratio as I-B ratio. We compute this I-B ratio for different values of the log ratio as well as the minimum support. Fig.4 illustrates the results. As shown in Fig.4 (a), with decrease in the log ratio, the I-B ratio falls down. This means that the more the non-executable sequences, the more useless effort is taken in basic algorithm. The exception in the point 0.6 arises from the fact that the log is generated randomly and it is possible that in some logs the number of the non-executable candidate sequences be at a low level which causes a reduction in the useless effort. Fig.4 (b) states another interesting point. Although we save more time in case of the lower minimum supports in the same log, but the I-B ratio remains constant approximately. The reason is that the increase and decrease of the minimum support causes the decrease and increase respectively on the number of the candidate sequences in both algorithms and this same effect keeps the ratio unchanged.

In summary, we observe that application of our idea for mining the frequent executable sequence yields a great benefit and causes great saving in the time. The experiment shows the average value of 0.23 for I-B ratio over all runs including the different log ratio and different minimum supports. This means that we save %77 of the time in average and this is actually a great percent. Taken into account different log ratios, in best case we gain %87 saving in 0.1 and %61 in worst case for 0.9 which still is a large amount even though only 0.1 of the log contains non-executable sequences.

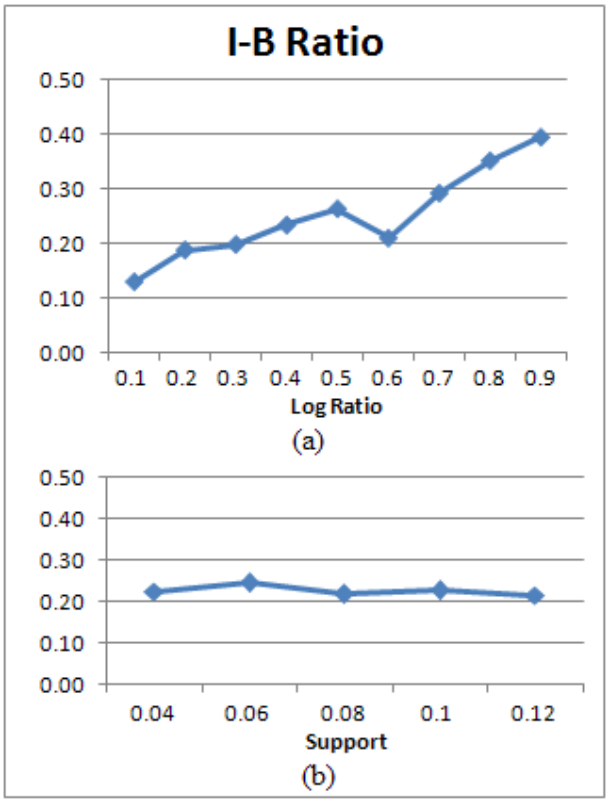

Fig.4. Execution time of improved algorithm over Execution time of base algorithm. (a) based on different log ratios. (b) based on different minimum supports.

6 Related Work

Noticeable efforts have been made in order to apply data mining techniques in the web known as the web mining. Nowadays, due to the increasing use of the web services, there is a tendency to apply those techniques in the field of the web service and this has made the web service mining a hot topic for research. Q. A. Liang et al. [2] introduced the term service mining and used it to compose web services. R. Gombotz et al. [7] have introduced the concept of Web Service Interaction Mining (WSIM) as mining patterns in the interactions of the web services with each other. They defined three levels of abstraction regarding WSIM: the operation level, the interaction level and the workflow level. They also provided appropriate web service log format in each of these three levels. M. Ruached et al. [9] used web service log mining to enable the verification of the behavioral properties in web service composition. They also proposed a logging solution to collect web service usage data. Q. A. Liang et al. [10] have introduced the concept of Service Usage Mining as the pattern discovery through web service usage data. They defined service usage data at three different levels: user request

level, template level and instance level. In their paper, an algorithm for service association rule mining in template level is provided but other algorithms in other levels aren't presented and are planned for future work. Additionally, defining a service-pattern-discovery enabled registry-repository architecture is another contribution of that work.

7 Conclusion and Future Works

The paper focuses on web service usage mining as the task of discovering frequent sequential pattern from the web service usage log. A format for the web service usage log was provided and the *AprioriAll* algorithm was proposed as a basic algorithm for mining sequential patterns according to the presented log format. The paper further focused on a special kind of the sequence of the web service execution and introduced the concept of executable sequence. In order to mine sequential patterns in executable sequences –so called executable sequential pattern mining- some modifications were applied to basic algorithm to achieve better performance. Finally, the performance of improved algorithm was investigated using some experiments over an analytical log. The results showed the average gain of %77 in execution time. This amount grows up with the decrease in the ratio of the number of the executable sequences over the non-executable sequences in the usage log.

In the future, we are intended to investigate the application of the concept of the executable sequences in web service composition and to develop a web service composition method which exploits the executable sequential pattern analysis algorithm.

References

[1] R. Nayak, and C. Tong, Applications of data mining in web services, *Lecture Notes in Computer Science*, Vol. 3306, November 2004, pp. 199-205.

[2] Q. Liang, S. Miller, and J. Chung, Service mining for web service composition, *IEEE International Conference on Information Reuse and Integration (IEEE IRI-2005)*, Las Vegas, Nevada, USA, 2005.

[3] R. Agrawal, and R. Srikant, Mining sequential patterns, *Eleventh International Conference on Data Engineering, IEEE Computer Society Press*, Taipei, Taiwan, 1995, pp. 3-14.

[4] R. Cooley, B. Mobasher, and J. Srivastava, Web Mining: Information and Pattern Discovery on the World Wide Web, *Proceeding of 9th IEEE Intl. Conf. on Tools with AI*, 1997.

[5] W. Tong, and H. Pi-Lian, Web log mining by an improved AprioriAll algorithm, *WEC'05: The Second World Enformatika Conference*, 2005.

[6] S. Dustdar, and W. Schreiner, A survey on web services composition, *International Journal of Web and Grid Services*, Vol. 1, 2005, pp. 1–30.

[7] J. Rao, and X. Su, A survey of automated web service composition methods, *Proceeding of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC2004)*, LNCS, San Diego, USA, 2004.

[8] R. Gombotz, K. Baina, and S. Dustdar, "Towards web services interaction mining architecture for e-commerce applications analysis," *Proceedings of the Conference on E-Business and E-Learning*, 2005.

[9] M. Rouached, W. Gaaloul, Wil M. P. van der Aalst, S. Bhiri, and C. Godart, "Web Service Mining and Verification of Properties: An Approach Based on Event Calculus**,"** *Cooperative Information Systems (CoopIS) 2006 International Conference, Lecture Notes in Computer Science*, 4275: pp 408-425, 2006.

[10] Q. A. Liang, J. Y. Chung, S. Miller, and Y. Ouyang, "Service pattern discovery of web service mining in web service registry-repository," *IEEE International Conference on e-Business Engineering (ICEBE'06)*, 2006.