Computers, Materials & Continua CMC, vol.63, no.3, pp.1221-1235, 2020

Prediction of Web Services Reliability Based on Decision Tree Classification Method

Zhichun Jia[1, 2], Qiuyang Han[1], Yanyan Li[1], Yuqiang Yang[1] and Xing Xing[1, 2, *]

Abstract: With the development of the service-oriented computing (SOC), web service has an important and popular solution for the design of the application system to various enterprises. Nowadays, the numerous web services are provided by the service providers on the network, it becomes difficult for users to select the best reliable one from a large number of services with the same function. So it is necessary to design feasible selection strategies to provide users with the reliable services. Most existing methods attempt to select services according to accurate predictions for the quality of service (QoS) values. However, because the network and user needs are dynamic, it is almost impossible to accurately predict the QoS values. Furthermore, accurate prediction is generally time-consuming. This paper proposes a service decision tree based post-pruning prediction approach. This paper first defines the five reliability levels for measuring the reliability of services. By analyzing the quality data of service from the network, the proposed method can generate the training set and convert them into the service decision tree model. Using the generated model and the given predicted services, the proposed method classifies the service to the corresponding reliability level after discretizing the continuous attribute of service. Moreover, this paper applies the post-pruning strategy to optimize the generated model for avoiding the over-fitting. Experimental results show that the proposed method is effective in predicting the service reliability.

Keywords: Decision tree, reliability level, quality of service, continuous attribute.

1 Introduction

Web services provide a standard means of communication among the different software applications on Internet. With the increasing number of web services with the same function, the application process requires qualitative selection of automatic service candidates [Yu, Zhang and Lin (2007)]. The reliable running of service process depends on both functional and nonfunctional service attributes, which are highly influenced by the quality of the selected web service [Silic, Delac and Srbljic (2013)]. To get an efficient service application, the service selection process should evaluate the reliability of web services according to both functional and nonfunctional service attributes [Avizienis, Laprie, Randell et al. (2004)]. The real relia bility of web service is affected

[1] College of Information Science and Technology, Bohai University, Jinzhou, 121013, China.

[2] School of Engineering, The University of New Mexico, Albuquerque, NM 87131, USA.

[*] Corresponding Author: Xing Xing. Email: xingxing@bhu.edu.cn.

Received: 16 January 2020; Accepted: 24 February 2020.

by many service attributes, such as response time, availability and throughput. Using the different evaluating attributes may result in a totally different reliability to the same web service, even in the same evaluating method. In addition, adopting mass attributes in the evaluating process results in the increasing of the computing time, which eventually is bound to reduce the effectiveness of evaluating results. Hence, how to quickly predict the service reliability before or during the application execution becomes a big challenge.

The researchers have proposed some definitions about the service reliability. The traditional definition of the service reliability is system-centric, such as availability, successability [Lyu (1996)]. Some literatures present the user-centric definitions, such as user-perceived reliability [Wang and Trivedi (2009)], or reliability on demand [Cortellessa and Grassi (2007)]. However, it should be noted that the service invocations are discrete and relatively sparse events. So the user-centric definition is more suitable for web services. Based on the above analysis, we focus on the prediction for the user-centric reliability in this paper. We define service reliability as the probability that the service invocation is successfully received under the specified conditions and time constraints. At present, there are a variety of different prediction methods for web services quality [Wang, Wang and Ding (2013); Silic, Delac, Krka et al. (2014); Wu, Xu, Lu et al. (2015); Xu, Zheng and Lyu (2015); Zheng, Trivedi, Qiu et al. (2015); Honamore, Dev and Honmore (2016); Wang, Wang, Yu et al. (2016); Yu and Huang (2016); Kumar and Sureka (2017); Thinh (2017); Wang, Yang and Yu (2017); White, Palade and Clarke (2017); Chen and Ha (2018)]. The most popular methods for predicting reliability of web services are mainly based on the collaborative filtering technology [Zheng and Lyu (2010); Zheng, Ma, Lyu et al. (2010); Yu and Huang (2016); Wu, Zhang, Luo et al. (2018); Wang, Chen, Shang et al. (2019)]. The advantage of this technology is the service reliability can be predicted, even if we cannot get some important prediction data. However, many other technologies are also used in the service prediction, such as k-means clustering [Su and Khoshgoftaar (2009); Maamar and Benahmed (2019); Yang, Zhou and Yang (2019)], Markov [Almulla, Almatori and Yahyaoui (2011); Zheng, Trivedi, Qiu et al. (2015)], HMM [Rahnavard, Najjar and Taherifar (2010); Honamore, Dev and Honmore (2016)], graph reduction [Cardoso, Sheth, Miller et al. (2004)], and fuzzy logic [Almulla, Almatori and Yahyaoui (2011); Honamore and Rath (2016)].

In this paper, we present a reliability prediction model based on the decision tree, and propose a prediction algorithm. The advantage of the decision tree technology is we can get the minimal and optimal attribute set for evaluating the reliability of web service by selection of the optimal split attributes and pruning process, and reduce the evaluating time. Our method applies the quality of web services to construct the service classification tree. By discretizing the continuous attribute of service, an optimum reliability model for web services is built and can quickly predict the reliability level of the given service.

The rest of paper is organized as follows. In Section 2, we describe the basic construction process of the decision tree. In Section 3, we present our reliability prediction process for web services by extracting the service data from the net-work. In Section 4, we propose our reliability prediction model for web services. In Section 5, we evaluate the proposed method by QWS database and show our experimental results. Finally, we present our

conclusions in Section 6.

2 Construction of decision trees

A decision tree is a representation of a decision procedure for determining the class of a given instance [Utgoff (1989)]. Generally, a decision tree consists of three kinds of node: root node, child node and leaf node. Each node of the tree specifies either a class name or a specific test that partitions the space of instances at the node according to the possible outcomes of the test [Kaur and Kaur (2019)]. A root node includes the universal set of instances. A leaf node denotes a classification result. Each child node represents a test for one attribute. Each subset of the partition corresponds to a classification sub-problem for that subspace of the instances, which is solved by a sub-tree. A decision tree can be seen as a divide-and-conquer strategy for object classification [Utgoff (1989)]. The decision tree is built recursively according to a given training set. In each recursion step, an instance is divided into a sub-tree with maximum similarity. And the current node is expanded recursively on each of all subsets of the training set which are defined by the instance attributes. A basic construction algorithm of decision tree is described in Tab. 1.

Table 1: Construction algorithm of decision tree

Input: training set $D=\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_n,y_n)\}$, attribute set $A=\{a_1,\ldots,a_m\}$

Output: a decision tree ***CDTree*** and N which is the root node

```
to generate a node N;
if ∀y_i ∈ C, 1 ≤ i ≤ n then
    to mark N as a leaf node of C;
    return;
end if
if A = ∅ or any attribute of any two x had the same value then
    to mark N as a leaf node of the class with maximum number in D;
    return;
end if
select the optimal split attribute a_* from A;
for each value a_*^v in a_* do
    to generate a sub-tree of N;
    D_v ⊆ D and ∀d ∈ D_v, a_*^d = a_*^v;
    if D_v = ∅ then
        to mark the sub-tree of N as a leaf node of the class with maximum number in
        return;
    else
        to mark CDTree(D_v, A \ {a_*}) as a sub-tree;
    end if
end for
```

In Tab. 1. The D is the training set, and each training instance is denoted by a set of attribute-value pairs $\boldsymbol{x}_i$ and a class label y_i. The A is a set of attributes which describes an instance in D. For each attribute a_j in A, a_j^v indicates the possible value of a_j. The *CDTree* is a decision tree function, and it returns a decision tree according to the given training set.

In the construction algorithm of decision tree, the most critical issue is how to get the goodness of split measure. Generally, there are three measure to be used in selecting the optimal split attributes. They include the information gain, gain ratio and Gini index [Mingers (1989)]. In this paper, we use the Gini index to measure the optimal split attribute.

3 Reliability prediction process

As mentioned before, we focus on the prediction of web service reliability according to the given training set. In Fig. 1, we present the prediction process of service reliability.

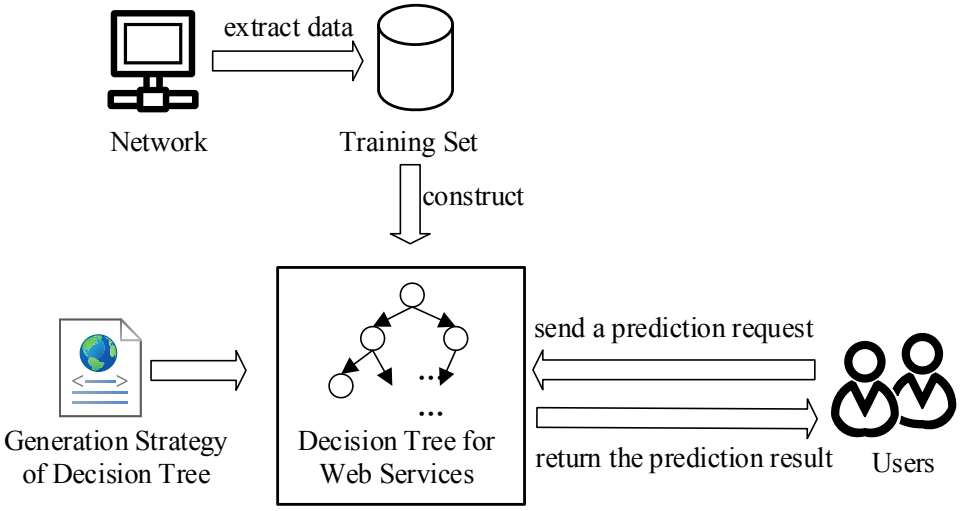

Figure 1: Prediction process for the web service reliability

As can be seen in Fig. 1, reliability prediction process is consisted of three phases: data extracting phase, constructing decision tree phase and prediction phase. Prior to construct decision tree, we perform collecting the quality data of web services from the network. Then, we use the generated training set and the given generation strategy of decision tree to construct the decision tree for web services. Finally, when the user sends a reliability prediction request, the prediction result of the reliability can be generated by the decision tree for web services and return to the user.

4 Prediction model

This section describes each step of constructing the service decision tree and defines how reliability prediction are calculated from the reliability level.

A service training set is formally defined as follows:

$$S = \{(\boldsymbol{s}_1, r_1), \ldots, (\boldsymbol{s}_n, r_n)\} \tag{1}$$

where $\boldsymbol{s}_i$ is a web service and $1 \le i \le n$, and r_i is the reliability level of the service $\boldsymbol{s}_i$.

A service attribute set is formally defined as follows:

$$QoS = \{qs_1, \ldots, qs_m\} \tag{2}$$

where qs_j is a service attribute of web services, such as response time, availability.

4.1 Reliability levels

In here, we define five reliability levels $\boldsymbol{R} = \{R_1, R_2, R_3, R_4, R_5\}$ in this paper, which are shown in Tab. 2.

Table 2: Reliability levels

Reliability Levels	Reliability Range
R_1	[0.9, 1.0]
R_2	[0.8, 0.9)
R_3	[0.7, 0.8)
R_4	[0.6, 0.7)
R_5	[0.0, 0.6]

4.2 Select the optimal split attribute

In this paper, we use the Gini index to select the optimal split attribute, and the services are partitioned into subset according to the values of that attribute.

The Gini index is proposed by Breiman et al. [Breiman, Friedman, Stone et al. (1984)] in 1984. The Gini function measures the 'impurity' of an attribute with respect to the classes [Mingers (1989)]. The Gini function is formally defined as follows:

$$Gini(S) = \sum_{k=1}^{|\boldsymbol{R}|} \sum_{k' \ne k} p_k p_{k'} = 1 - \sum_{k=1}^{|\boldsymbol{R}|} p_k^2 \tag{3}$$

where $|\boldsymbol{R}|$ is the number of reliability levels, p_k is the probability of the kth reliability level services in the set *S*.

The Gini Index method was used to split off the largest category into a separate group, with the default split size set to enable growing the tree. In this case, we estimate the reliability level probabilities with the actual relative frequencies. Thus the impurity of the attribute qs_j is as follows:

$$Gini\_index(S, qs_j) = \sum_{v=1}^{V_j} \frac{|S^{v_j}|}{|S|} Gini(S^{v_j}) \tag{4}$$

where V_j denotes the number of different value of the attribute qs_j in S, $|S|$ denotes the number of instances in S, $|S^{v_j}|$ denotes the number of instances with value v_j of attribute qs_j in S. The smaller the Gini index is the higher the purity of S. Therefore, we select the attribute qs_* with minimum value of the Gini index as the split attribute.

$$sq_* = \arg\min_{1 \le j \le m} Gini\_index(S, qs_j) \tag{5}$$

4.3 Dealing with continuous value

In the quality of web service, many attributes are the continuous attributes. Thus we use the bi-partition method to deal with them in this paper.

If the attribute qs_j is a continuous attribute, we first use the Quicksort ordering algorithm to order the instances from small to large according to the values of qs_j. Assume that the ordered values are $v_1, \ldots, v_m$. Consider for $j \in [1, m-1]$, the value $v = (v_j + v_{j+1})/2$ and the splitting is as follows:

$$S_1^v = \{v_k \mid v_k \le v\} \tag{6}$$

$$S_2^v = \{v_k \mid v_k > v\} \tag{7}$$

For each value v, the Gini index is computed by considering the splitting above. The value v' for which $Gini\_index(S, qs_j, v')$ is minimum is set to be the classification threshold of the attribute qs_j.

4.4 Prediction algorithm

According to the above description, we present our prediction process in Tabs. 3 and 4.

The service construction algorithm is a construction procedure of classification way for the service reliability level according to the given service set. In Tab. 3, the *S* is the service set, and each service is marked by a reliability level sign. The *QoS* is a set of attributes which describes a service in *S*. The *ST* is a service decision tree, and it returns a service decision tree according to the given service set *S* and attribute set *QoS*. The function ComputeClassFrequence() counts the service number of each reliability level in the service set *S*. The OneReliabilityLevel denotes that there is one reliability level in the service set *S*. The FewServices means that the service set *S* is a null set.

Table 3: Service construction algorithm

Input: service set $S = \{(s_1, r_1), \ldots, (s_n, r_n)\}$,
attribute set $QoS = \{qs_1, \ldots, qs_m\}$
Output: a service decision tree ***ST***

```
01: ComputeClassFrequence(S);
02: if OneReliabilityLevel or FewServices then
03:     return;
04: end if
05: create a decision node N;
06: if attribute qs is continuous then
07:     find classification threshold;
08: end if
09: for each attribute qs do
10:     ComputeGiniIndex(S);
11: end for
12: N = AttributeWithSmallerGiniIndex;
13: for each S' in the splitting of S
14:     if S' = ∅ then
15:         Child of N is a leaf;
16:     else
17:         Child of N = ST(S', QoS);
18:     end if
19: end for
```

The level prediction algorithm is a prediction process to the given service using above generated service decision tree in the service construction algorithm. In Tab. 4, the *ST* is a service decision tree, and the level prediction algorithm can classify the given service s' by the function ClassficationService () and *ST*.

Table 4: Level prediction algorithm

Input: service set $S = \{(s_1, r_1), \ldots, (s_n, r_n)\}$,
attribute set $QoS = \{qs_1, \ldots, qs_m\}$,
predicted service s'
Output: the reliability level of s';

```
01: ST(S, QoS);
02: ClassficationService ( s', ST);
03: return the reliability level of s';
```

4.5 Pruning

In order to avoid the over-fitting of the service decision tree, we use the service tree pruning algorithm in Tab. 5 to obtain a right sized tree. Firstly, we build a complete service decision tree by the service construction algorithm in Tab. 3. Then, we remove sub-trees that are not contributing significantly towards generalization accuracy.

In the first stage, a sequence of increasingly smaller trees is built on the training data. In the second stage, one of these trees is chosen as the pruned tree, based on its classification accuracy on a pruning set.

Table 5: Service pruning algorithm

Input: running service set $S_p = \{(\boldsymbol{s}_1, r_1), \ldots, (\boldsymbol{s}_n, r_n)\}$,

pruning attribute set $QoS_p = \{qs_1, \ldots, qs_m\}$,

the decision tree ***ST***,

Output: a service pruning tree ***SP***

```
01: testTree=ST; SP=ST;
02: while each node in ST
03:    deleteLeaf(testTree);
04:                  if    ComputeAccuracy(testTree)    >
05:       SP=testTree;
06:    end if
07:    testTree=SP;
08: end while
```

5 Experiments

In this section, we present the evaluate results that supports our claims that our reliability prediction model improves the prediction accuracy. To prove our claims, we use the QWS database [Al-Masri and Mahmoud (2007)] to conduct the experiments and analyze the results. The QWS database collects 2507 real web services, and each service is described by 11 parameters including service name, description, operation name, description, message name, and message description tags. Tab. 6 shows 8 attributes of web service we used in the experiments.

Table 6: Attributes in QWS

Attributes	Definition
Response Time	Time taken to send a request and receive a response
Availability	Number of successful invocations/total invocations
Throughput	Total Number of invocations for a given period of time
Successability	Number of response/number of request messages
Compliance	The extent to which a WSDL document follows WSDL specification
Best Practices	The extent to which a Web service follows WS-I Basic Profile
Latency	Time taken for the server to process a given request
Documentation	Measure of documentation (i.e., description tags) in WSDL

To evaluate prediction accuracy for web service, we use the standard error measures: mean absolute error MAE. The MAE represents the average magnitude of errors for the predicted reliability values:

$$MAE = \frac{\sum_j^N \left| p_j - \hat{p}_j \right|}{N} \tag{8}$$

where N is the cardinal number of the prediction set, p_j real reliability level is from QWS, while $\hat{p}_j$ is the predicted reliability level.

In our experiments, we select 1500 web services from QWS as the training set, 500 services as the pruning set, and 500 services left are used as test set. Firstly, we separately use the 500, 1000 and 1500 training data to generate the service decision trees. Then, we use 500 pruning data to separately prune the three service decision trees, and get the pruned service trees. The pruned service tree by 500 training data includes three split attributes: best practices, compliance and availability. The pruned service tree by 1000 training data includes three split attributes: best practices, availability, documentation and response time. The pruned service tree by 1500 training data includes the same attributes with the second pruned service tree. Next, we use the above two split attribute sets ('best practices, compliance, availability' & 'best practices, availability, documentation, response time') to generate the service decision trees and pruned service trees. At the same time, we get two split attribute sets ('best practices, availability' & 'best practices') according to the pruned trees. We use the two split attribute sets again to generate the new service decision trees. Finally, we compare the accuracy, MAE and running time of the five groups with different split attribute sets and the number of training data.

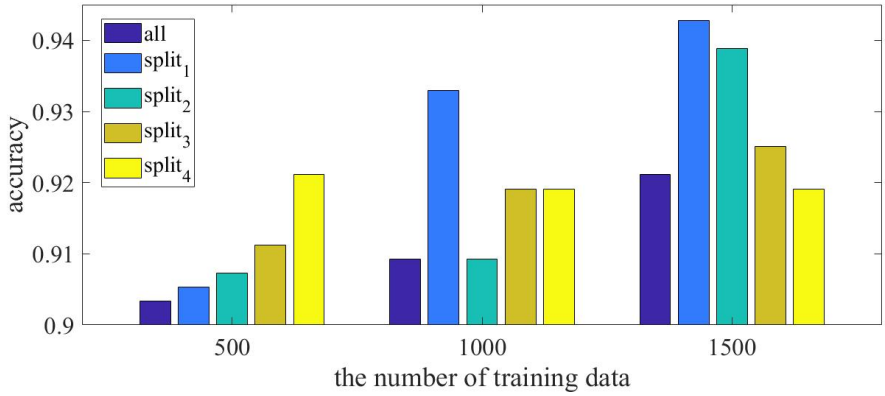

Figure 2: Accuracy comparison with the different split attributes

Fig. 2 shows the comparison results of the prediction accuracy with different split attribute sets and the number of training data. In Fig. 2, 'all' denotes that the split attribute set includes all attributes in Tab. 2; '$split_1$' denotes that the split attribute set includes best practices, compliance and availability; '$split_2$' denotes that the split attribute set includes best practices and availability; '$split_3$' denotes that the split attribute set includes best practices, availability, documentation and response time; '$split_4$' denotes

that the split attribute set includes best practices. When we consider all attributes as the split attribute set, the prediction accuracies are the lowest in three groups of test sets. The prediction accuracies of '$split_1$' are the highest in the second and third groups. As shown in the figure, when the training data is above certain level, the split attribute set '$split_1$' can get the higher prediction accuracy. This is because best practices, compliance and availability contain more useful information for evaluating the reliability of web service. Without the attribute 'compliance', this leads to the worse performance when adopting the split attribute set '$split_2$' to predict the service reliability. Moreover, the execution results of 'all', '$split_3$' and '$split_4$' are not ideal due to the lack of useful attributes or the adoption of disturbed attributes.

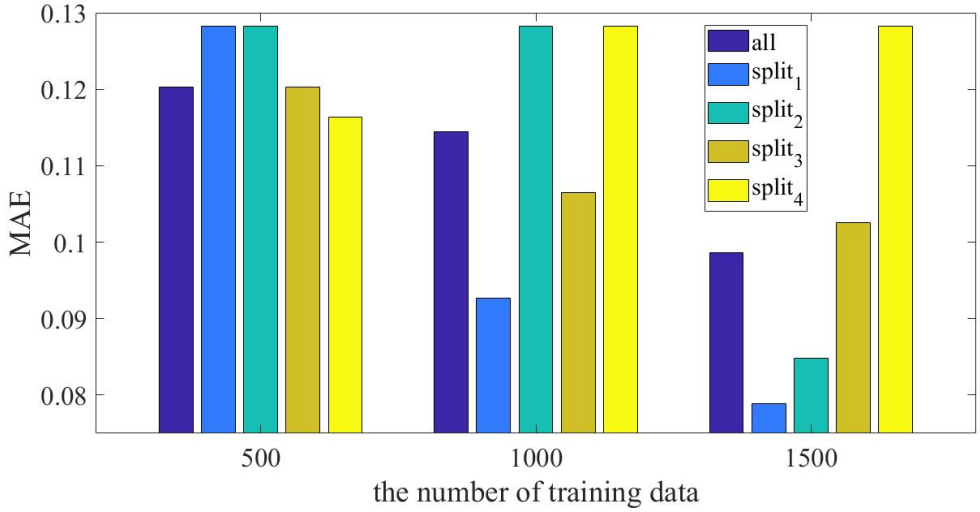

Figure 3: MAE comparison with the different split attributes

Fig. 3 shows the comparison results of the mean absolute error with different split attribute sets and the number of training data. The MAE of '$split_1$' are the lowest in the second and third groups. Consequently, when the training data is above certain level, the split attribute set '$split_1$' can get the lower mean absolute error. This result again demonstrates that, the attributes 'best practices, compliance and availability' are closely related with the reliability of service.

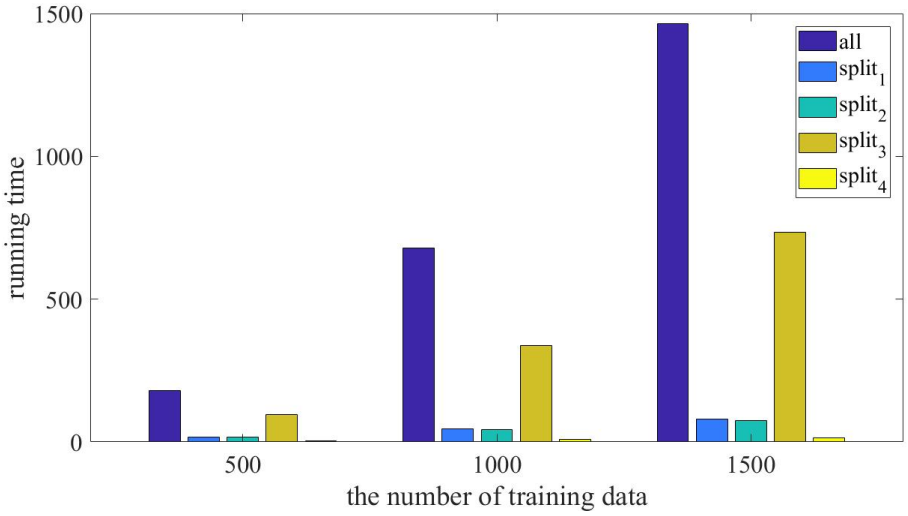

Figure 4: Running time comparison with the different split attributes

Fig. 4 shows the comparison results of the mean absolute error with different split attribute sets and the number of training data. The running time of 'all' is the longest in three groups. The running time of '$split_4$' is the shortest. Consequently, the number of attributes in the split set is positively related to the running time. The more the attribute number is, the longer the running time is. By contrast, the less the attribute number is, the shorter the running time is.

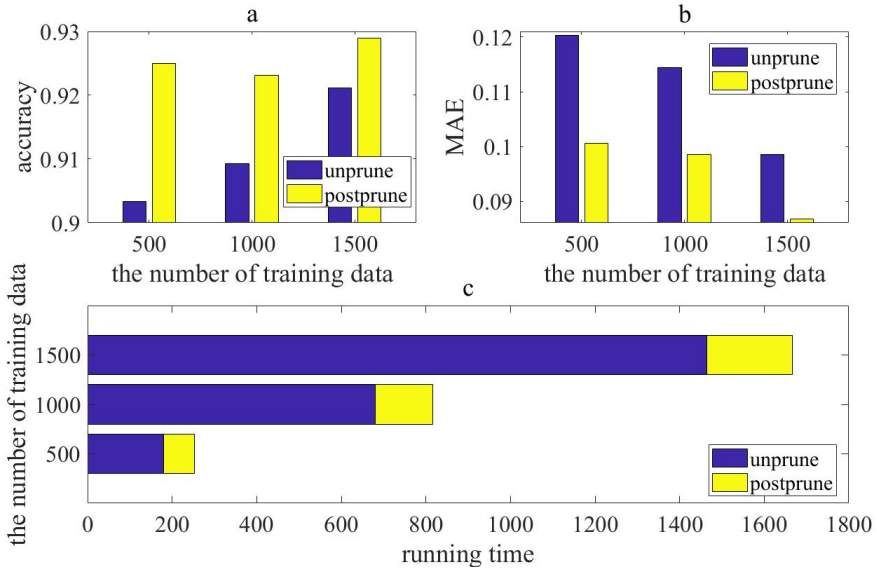

Figure 5: Un-pruning and post-pruning comparison with all attributes

Fig. 5(a) shows the accuracy comparison result of the un-pruning and post-pruning with 'all' split attribute set. Fig. 5(b) shows the MAE comparison result of the un-pruning and

post-pruning with 'all' split attribute set. We can see that the accuracy is increased appreciably from Fig. 5(a), and the MAE is reduced appreciably from Fig. 5(b). Fig. 5(c) shows the running time comparison result of the un-pruning and post-pruning with 'all' split attribute set. We can see that the running time has a rapid growth with the increasing number of training data. Fig. 5 explains the significance of pruning. We can achieve the closely related attribute set, improve the performance result, and reducing the running time.

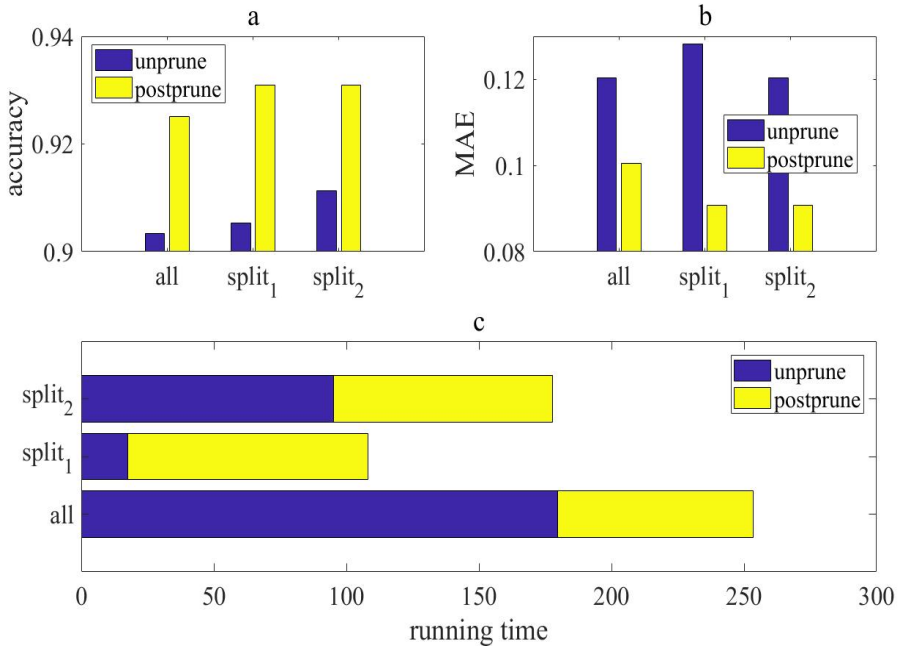

Figure 6: Un-pruning and post-pruning comparison with the different split attributes

Fig. 6(a) shows the accuracy comparison result of the un-pruning and post-pruning with the different split attributes. Fig. 6(b) shows the MAE comparison result of the un-pruning and post-pruning with the different split attributes. Similar to Fig. 5, the three accuracies with different split sets are increased appreciably after the pruning, and the three MAEs are reduced appreciably after the pruning. Fig. 6(c) shows the running time comparison result of the un-pruning and post-pruning with the different split attributes. We can see that the running time of '$split_1$' is shortest from Fig. 6(c).

As the figures show, our method is effective for the reliability prediction of web services. Our experimental results show that we can get the higher prediction accuracy and the shorter running time when we use the attributes 'best practices, compliance and availability' as the split attribute set.

6 Conclusions

Aiming at the dynamic of the network and user needs, this paper proposes a prediction approach of the service reliability level by constructing the service decision tree. We utilize the decision tree method to derive a reliability prediction model that classifies

services into the corresponding reliability level according to the related attribute values. A selection method of optimal split attribute and a bi-partition method for dealing with the continuous attribute are presented to support the reliability prediction model. To solve the over-fitting problem of the model and improve the prediction accuracy, we exploit a service tree post-pruning method. The experiments based on a real-world dataset reveal that the proposed method provide a high accuracy of prediction without noticeably increased running time. The proposed method contributes to reliability prediction in a high-dynamic computing environment.

Funding Statement: This paper is partially supported by the National Natural Science Foundation of China under Grant No. 61972053 and No. 61603054, by the Scientific Research Foundation of Liaoning Education Department under Grant No. LQ2019016, No. LJ2019015, and by the Natural Science Foundation of Liaoning Province, China under Grant No. 2019-ZD-0505.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Al-Masri, E.; Mahmoud, Q. H. (2007): Discovering the best Web service. *Proceedings of the 16th International Conference on World Wide Web*, pp. 1257-1258.

Almulla, M.; Almatori, K.; Yahyaoui, H. (2011): A QoS-based fuzzy model for ranking real world web services. *IEEE International Conference on Web Services*, pp. 203-210.

Avizienis, A.; Laprie, J. C.; Randell, B.; Landwehr, C. (2004): Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable Secure Computing*, vol. 1, no. 1, pp. 11-33.

Breiman, L.; Friedman, J.; Stone, C. J.; Olshen, R. A. (1984): Classification and regression trees. CRC press, New York.

Cardoso, J.; Sheth, A.; Miller, J.; Arnold, J.; Kochut, K. (2004): Quality of service for workflows and Web service processes. *Web Semantics: Science*, *Services Agents on the World Wide Web*, vol. 1, no. 3, pp. 281-308.

Chen, L.; Ha, W. (2018): Reliability prediction and QoS selection for web service composition. *International Journal of Computational Science Engineering*, vol. 16, no. 2, pp. 202-211.

Cortellessa, V.; Grassi, V. (2007): *Reliability Modeling and Analysis of Service-oriented Architectures. Test and Analysis of Web Services*. Springer, Berlin, Heidelberg.

Honamore, S.; Dev, K.; Honmore, R. (2016): Reliability prediction of web services using HMM and ANN models. *Procedia Computer Science*, vol. 93, pp. 41-47.

Honamore, S.; Rath, S. K. (2016): A Web service reliability prediction using HMM and fuzzy logic models. *Procedia Computer Science*, vol. 93, pp. 886-892.

Kumar, L.; Sureka, A. (2017): Neural network with multiple training methods for Web service quality of service parameter prediction. *Tenth International Conference on Contemporary Computing*, pp. 1-7.

Lyu, M. R. (1996). *Handbook of Software Reliability Engineering*. IEEE computer society press, CA.

Maamar, A.; Benahmed, K. (2019): A hybrid model for anomalies detection in AMI system combining k-means clustering and deep neural network. *Computers*, *Materials & Continua*, vol. 60, no. 1, pp. 15-39.

Mingers, J. (1989): An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, vol. 3, no. 4, pp. 319-342.

Rahnavard, G.; Najjar, M. S.; Taherifar, S. (2010): A method to evaluate Web services anomaly detection using hidden Markov models. *International Conference on Computer Applications and Industrial Electronics*, pp. 261-265.

Silic, M.; Delac, G.; Krka, I.; Srbljic, S. (2014): Scalable and accurate prediction of availability of atomic web services. *IEEE Transactions on Services Computing*, vol. 7, no. 2, pp. 252-264.

Silic, M.; Delac, G.; Srbljic, S. (2013): Prediction of atomic web services reliability based on k-means clustering. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pp. 70-80.

Su, X.; Khoshgoftaar, T. M. (2009): A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, vol. 2009, pp. 1-12.

Thinh, L. V. (2017): QoS prediction for web services based on restricted boltzmann machines. *Journal of Service Science Research*, vol. 9, pp. 197-217.

Utgoff, P. E. (1989): Incremental induction of decision trees. *Machine Learning*, vol. 4, no. 2, pp. 161-186.

Wang, D.; Trivedi, K. S. (2009): Modeling user-perceived reliability based on user behavior graphs. *International Journal of Reliability*, *QualitySafety Engineering*, vol. 16, no. 4, pp. 303-329.

Wang, H.; Wang, L.; Yu, Q.; Zheng, Z.; Bouguettaya, A. et al. (2016): Online reliability prediction via motifs-based dynamic Bayesian networks for service-oriented systems. *IEEE Transactions on Software Engineering*, vol. 43, no. 6, pp. 556-579.

Wang, H.; Wang, S.; Ding, F. (2013): Reliability prediction-based web service selection. *Advances in Information Sciences Service Sciences*, vol. 5, no. 9, pp. 391-399.

Wang, H.; Yang, Z.; Yu, Q. (2017): Online reliability prediction via long short term memory for service-oriented systems. *IEEE International Conference on Web Services*, pp. 81-88.

Wang, Q.; Chen, M.; Shang, M.; Luo, X. (2019): A momentum-incorporated latent factorization of tensors model for temporal-aware QoS missing data prediction. *Neurocomputing*, vol. 367, pp. 299-307.

White, G.; Palade, A.; Clarke, S. (2017): Qos prediction for reliable service composition in IoT. *International Conference on Service-Oriented Computing*, pp. 149-160.

Wu, F.; Xu, J.; Lu, J.; Xiao, G.; Zhang, Y. (2015): Real-time environment aware web service selection and evaluation. *Tenth International Conference on Digital Information Management*, pp. 198-203.

Wu, H.; Zhang, Z.; Luo, J.; Yue, K.; Hsu, C. H. (2018): Multiple attributes QoS prediction via deep neural model with contexts. *IEEE Transactions on Services Computing*, pp. 1-13.

Xu, J.; Zheng, Z.; Lyu, M. R. (2015): Web service personalized quality of service prediction via reputation-based matrix factorization. *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 1-10.

Yang, Y.; Zhou, D.; Yang, X. (2019): A multi-feature weighting based k-means algorithm for MOOC learner classification. *Computers*, *Materials & Continua*, vol. 59, no. 2, pp. 625-633.

Yu, C.; Huang, L. (2016): A Web service QoS prediction approach based on time- and location-aware collaborative filtering. *Service Oriented Computing Applications*, vol. 10, no. 2, pp. 135-149.

Yu, T.; Zhang, Y.; Lin, K. J. (2007): Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Transactions on the Web*, vol. 1, no. 1, pp. 1-26.

Zheng, Z.; Lyu, M. R. (2010): Collaborative reliability prediction of service-oriented systems. *ACM/IEEE 32nd International Conference on Software Engineering*, pp. 35-44.

Zheng, Z.; Ma, H.; Lyu, M. R.; King, I. (2010): Qos-aware Web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 140-152.

Zheng, Z.; Trivedi, K. S.; Qiu, K.; Xia, R. (2015): Semi-Markov models of composite web services for their performance, reliability and bottlenecks. *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 448-460.