Automated Fault Detection for Web Services using Naïve Bayes Approach

Xing Xing, Jianyan Luo, Zhichun Jia, Yanyan Li, and Qiuyang Han
College of Information Science and Technology
Bohai University
Jinzhou, Liaoning 121013, China
{xingxing & zhichun.jia}@bhu.edu.cn

Abstract—**With the development of the web service technology, the service application becomes an important and popular solution for the distributed computing model. As more and more web services are deployed on the network, the web services can fail for many reasons. One of challenges in this evolution is how to provide a necessary fault detection method for minimizing the service failure impact and enhance the reliability of the services. For this purpose, we present an automatic fault detection framework and a fault detection model based on the Naïve Bayes approach. By analyzing the available service execution logs, our method can achieve the training data and convert them into the detection model. Using the Naïve Bayes approach and the given threshold value, our model computes the service posterior probability to each fault type and identifies the service faults. Experimental results show that our method is effective in detecting the service faults.**

Keywords-Naïve Bayes; fault detection; service fault; execution log; fault type

I. INTRODUCTION

Web services provide a standard means of communication among the different software applications on Internet. With the increasing size and complexity of web services, the service processes have an amplified number of potential failures, and make them harder to ensure the service reliability [1]. Since the web services run in the highly dynamic environment, and the service itself may have inherent errors, it may occur various faults during the execution [2]. It is clearly impossible to avoid all the faults which can cause the poor performances for web services. Hence, how to quickly detect the service faults and correctly classify the faults before or during the service execution becomes a big challenge.

In general, the fault detection is recognizing that a problem has occurred, even if you don't yet know the root cause. Faults may be detected by a variety of quantitative or qualitative means. The fault diagnosis is pinpointing one or more root causes of problems, to the point where corrective action can be taken. Once the faulty type is detected or diagnosed, the service process can use a recovery mechanism to recover the process from the identified fault [3]. Traditional fault detection and diagnosis methods [4-21] on web services are well suited for handling fault of the business logic, but they are too sophisticated to get an exact result in a limited time. A QoS-supported approach [3] is used to detect and tolerate the fault in a service process. But this approach can handle only two faults, and cannot handle the network fault and incorrect value. A Bayes method [22] focuses on monitor and diagnose the software defined networks (SDN) using the technologies of machine learning and data analysis. However, it cannot cope the complex cases.

In this paper, we present an automated fault detection framework, and propose a fault detection method based on Naïve Bayes. Our method applies the Bayes' rule to compute the correlation coefficient between the service properties and fault. By the result ranking and given threshold value, our method can decide whether or not to fault occurs and which type the fault is.

The rest of paper is organized as follows. In Section 2, we describe the rules of Naïve Bayes. In Section 3, we present our automated fault detection framework for the web services. In Section 4, we propose our fault detection model. In Section 5, we conduct the simulation experiments and show our experimental results. Finally, we present our conclusions in Section 6.

II. NAÏVE BAYES

The Naïve Bayes approach is a simple probabilistic inference method with the conditional independence assumptions between propositions. A Naïve Bayes model is easy to build, and it often works surprisingly well, even when the conditional independent assumption is not true. So, it is widely used during to its high performance compared with more sophisticated probability methods.

$$\mathrm{P}(C \mid X) = \frac{\mathrm{P}(C)\prod_{j=1}^{m}\mathrm{P}(x_j \mid C)}{\mathrm{P}(X)} \tag{1}$$

The Naïve Bayes model includes the class variable C and attribute variable X. Given the class C, the Naïve Bayes model firstly assumes that the attributes X are conditionally independent of other attribute variables. Within a single pass to the training data, the model applies Bayes' theorem to classify new examples for which the class variable C is unobserved. With observed attribute values $x_1, \ldots, x_m$ (m is the number of

This paper is partially supported by the National Natural Science Foundation of China under Grant No.61603054, by the Scientific Research Foundation of Liaoning Education Department No. LQ2019016, No. LJ2019015, and by the Natural Science Foundation of Liaoning Province, China under Grant No.20170540016.

978-1-7281-0945-9/19/$31.00©2019 IEEE

observed attribute values), the probability of each class is given by equation (1).

In the equation (1), $1/P(X)$ is the normalization constant needed to make the entries in $P(C \mid X)$ sum to 1. So, we use the symbol α to replace $1/P(X)$. The equation (1) is transformed into the equation (2).

$$P(C \mid X) = \alpha P(C) \prod_{j=1}^{m} P(x_j \mid C) \quad (2)$$

In the equation (2), $P(C \mid X)$ is the posterior of class C given predictor X; $P(C)$ is the prior probability of class C; $\prod_{j=1}^{m} P(x_j \mid C)$ is the likelihood which is the probability of predictor X given class C. The Naïve Bayes includes all predictors using Bayes' rule and the independence assumptions between predictors.

III. Detection Framework

In this paper, we focus on the detection of long-running or computationally-intesive application services in the network. Due to the increasing complexity of service applications, softerware errors, malicious services and invalid inputs are quite likely. Automated fault detection model is thus proposed to ensure the safety and stability of the service system by quickly and exactly analyzing and finding faults.

For facilitating detection, we assume that the execution logs of service system are available and the all property values of system services are known. The execution log form of service system is as follow:

- The service execution log $SEL = \{sel_1, sel_2, \ldots, sel_n\}$ consists of n records, where each record sel_i $(1 \le i \le n)$ describes an execution of system;
- Each record $sel_i = \{sn_{i1}, sp_{i1}, \ldots, sp_{ik}, ft_i\}$ consists of the service name of the ith execution, k properties of the service and its fault type; ft_i shows the fault type of the ith execution; if $ft_i = \phi$, the ith is normal.

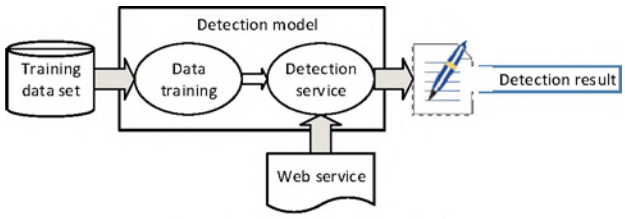

Figure 1. Detection framework

On the basis of the above assumptions, we propose an automated fault detection framework for web services in Fig. 1. Our detection framework mainly includes the following parts:

- Training data set: It is a set of all service execution logs which kept a record of each service running;
- Web service: It is a running web service which is monitored during the running;
- Detection model: It is responsible for training the data and detecting the service fault during the system running.

IV. Detection Model

Bayes classifier is classification model based on statistical theory. The Naïve Bayes classifier is popularly used for its simple structure and good performance. A Naïve Bayes classifier considers all of the service properties to independently contribute to the probability that a service is normal or fault.

A fault detection model for web services is a four-tuples $\mathbf{DM4W} = \{\boldsymbol{T}, \boldsymbol{S}, \boldsymbol{Q}, \boldsymbol{F}\}$. In the detection model, $\boldsymbol{T}$ is the training data set, $\boldsymbol{S}$ is the set of web services, $\boldsymbol{Q}$ is the set of the related properties of the web services, $\boldsymbol{F}$ is the set of fault type.

According to the Naïve Bayes' rule:

$$p(f_i \mid s_j) = \frac{p(f_i) \prod_{r=1}^{k} p(q_r \mid f_i)}{p(s_j)} \quad (3)$$

where s_j denotes a web service and $s_j \in \boldsymbol{S}$, q_r denotes the value of rth property Q_r of a web service and $Q_r \in \boldsymbol{Q}$, f_i denotes a fault type and $f_i \in \boldsymbol{F}$.

By the equation (3), we can get the probability of that the web service s_j occurs the fault f_i. According to the equation (2), we can use a normalization constant α to replace $1/p(s_j)$.

$$p(f_i \mid s_j) = \alpha p(f_i) \prod_{r=1}^{k} p(q_r \mid f_i) \quad (4)$$

Given a training data set $\boldsymbol{T}$, we can get the probability of that each service occurs every fault in $\boldsymbol{F}$ by the equation (4). A new web service is then classified by comparing the probability values. Finally, the proposed detection model achieves a detection result to users.

In the equation (4), the prior probability of given fault f_i is:

$$p(f_i) = N_{f_i} / N \quad (5)$$

where N_{f_i} is the sample amount of the fault f_i in $\boldsymbol{F}$, and N is the total number of all training data in $\boldsymbol{T}$.

Reckon the conditional probability of each service property by computing their likelihood probabilities. If the property Q_r is a discrete attribute, we use the equation (6) to compute the conditional probability:

$$p(q_r \mid f_i) = N_{f_i}^{q_r} / N_{f_i} \tag{6}$$

where $N_{f_i}^{q_r}$ denotes the amount of the training samples which satisfy $F = f_i$ and $Q_r = q_r$ in the training set $\boldsymbol{T}$.

In addition, if $N_{f_i}^{q_r} = 0$, we use the Laplacian correction to adjust the equation (5) and equation (6). The equation (5) is revised to equation (7):

$$\hat{p}(f_i) = \frac{N_{f_i} + 1}{N + n} \tag{7}$$

where n is the number of the fault types in the training set $\boldsymbol{T}$. The equation (6) is revised to equation (8):

$$\hat{p}(q_r \mid f_i) = \frac{N_{f_i}^{q_r} + 1}{N_{f_i} + N_{Q_r}} \tag{8}$$

where N_{Q_r} is the total number of different values of property Q_r in $\boldsymbol{Q}$.

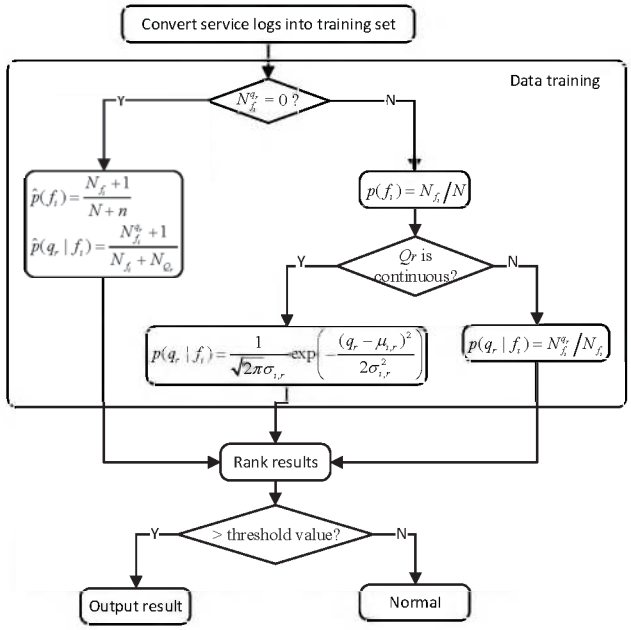

Figure 2. Algorithm flow

If the property Q_r is a continuous attribute, we consider the probability density function to compute the conditional probability, and assume $p(q_r \mid f_i) \sim \mathcal{N}(\mu_{i,r}, \sigma_{i,r}^2)$:

$$p(q_r \mid f_i) = \frac{1}{\sqrt{2\pi}\sigma_{i,r}} \exp\left(-\frac{(q_r - \mu_{i,r})^2}{2\sigma_{i,r}^2}\right) \tag{9}$$

where $\mu_{i,r}$ is the mean value of the property Q_r in the training samples $\boldsymbol{T}$ which satisfy $F = f_i$, $\sigma_{i,r}^2$ is the variance of the property Q_r in the training samples $\boldsymbol{T}$ which satisfy $F = f_i$.

Finally, the detection model gets the probabilities of all classifications. According to the probability values, we pick the one fault which has the maximum posterior probability to a service and the probability value is greater than the given threshold value. If no posterior probability to a service is greater than the given threshold value, we think the service cannot occur the fault. An algorithm flow is shown in Fig. 2.

V. EXPERIMENTS

To validate the effectiveness of the proposed detection method, we develop a simulation experiment environment to automatically generate the execution data of web services using the Matlab. Our simulation experiment environment shown in Fig. 3 consists of five components: QWS database [23], running data generator, training set, test set and detection model.

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. The QWS values represent averages of the measurements collected during that period.

The running data generator generates a set of the service execution logs using the QWS database. In addition, we inject three types of service faults to the service execution logs according to the related properties of the services. Three types of faults are network fault, server fault and sematic fault respectively.

Then, we pick 80 percent of the generated execution logs as the training set, and 20 percent of them as the test set.

Finally, the detection model computes the service error probabilities of three faults by the training set and the test service, and outputs the analytic result to users.

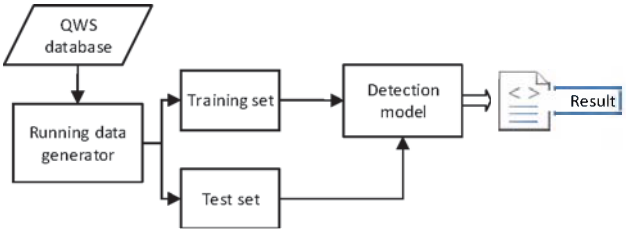

Figure 3. Simulation experiment environment

In our experiments, we use three metrics to evaluate the proposed method, including true positive, true negative and overall accuracy. The overall accuracy (OA) is not enough to determine the performance of any detection models. The true accuracy rate (TAR) and detection rate (DR) are equally important as measures for the performance of the fault detection models. Hence, we measure TAR and DR as well to check the performance of the proposed detection model. TAR is the percentage of the normal services which has been correct classified as normal by the equation (10). DR is opposite of

TAR, and is the percentage of the fault services which represents a correct classification as fault by equation (11). The OA is given by equation (12).

$$TAR = \frac{true\ positive}{false\ negative + true\ positive} \times 100\% \quad (10)$$

$$DR = \frac{true\ negative}{false\ positive + true\ negative} \times 100\% \quad (11)$$

$$OA = \frac{true\ positive + true\ negative}{number\ of\ test\ samples} \times 100\% \quad (12)$$

Here, a true positive is the number of the normal services which are correctly identified the normal services. Similarly, false negative is the number of the faulty services which are identified as normal services. The true negative is the number of the faulty services which are correctly identified faulty services. The false positive is the number of the normal services which are identified as faulty services.

TABLE I. DETECTION ACCURACY

Metrics	Rate
OA	83.5%
TAR	86.3%
DR	74.1%

As shown in Table 1, the detection accuracy of the proposed method is between 74-86% for the test set. And our method can provide the faulty type to users. Hence, the proposed method is effective for fault detection of web services.

VI. CONCLUSIONS

In this paper, we first present an automatic fault detection framework for the web services. In the framework, we use the service execution logs to get the training set and detect the service fault by the proposed detection model. The proposed model applies the Naïve Bayes approach to classifying the faulty service by the training set. According to the given threshold value, our model can decide whether or not to fault occurs and which type the fault is. Experimental results show that our method is effective to detect the service faults.

REFERENCES

[1] Z. Jia, R. Chen, and X. Xing, "Behavior similarity-based fault diagnosis for web services", Journal of Information & Computation Science, 10(17): 5699–5708, 2013.

[2] Z. Jia, R. Chen, X. Xing, J. Xu, and Y. Xie, "SFDCloud: top-k service faults diagnosis in cloud computing", Automated Software Engineering, 21(4): 461-488, 2014.

[3] R. Gupta, R. Kamal, and U. Suman, "A QoS-supported approach using fault detection and tolerance for achieving reliability in dynamic orchestration of web services", International Journal of Information Technology, 10(1): 71-81, 2018.

[4] S. Yin, G. Wang, and H. R. Karimi, "Data-driven design of robust fault detection system for wind turbines", Mechatronics, 24(4): 298-306, 2014.

[5] S. Yin, G. Wang, and X. Yang, "Robust PLS approach for KPI related prediction and diagnosis against outliers and missing data", International Journal of Systems Science, 45(7): 1375-1382, 2014.

[6] S. X. Ding, S. Yin, K. Peng, H. Hao, and B. Shen, "A novel scheme for key performance indicator prediction and diagnosis with application to an industrial hot strip mill", IEEE Transaction on Industrial Informatics, 9(4): 2239-2247, 2013.

[7] S. X. Ding, P. Zhang, S. Yin, E. L. Ding, "An integrated design framework of fault-tolerant wireless networked control systems for industrial automatic control applications", IEEE Transactions on Industrial Informatics, 9(1): 462-471, 2013.

[8] L. M. Elshenawy, S. Yin, A. S. Naik, and S. X. Ding, "Efficient recursive principal component analysis algorithms for process monitoring", Industrial & Engineering Chemistry Research, 49(1): 252-259, 2010.

[9] A. Liu, Q. Li, L. S. Huang, and M. J. Xiao, "FACTS: a framework for fault-tolerant composition of transactional web services", IEEE Transactions on Services Computing, 3(1): 46-59, 2010.

[10] Y. Yan, P. Dague, Y. Pencole, and M.-O. Cordier, "A model-based approach for diagnosing faults in web service processes", The International Journal of Web Services Research (JWSR), 6(1): 87-110, 2009.

[11] K.-L. Peng, and C.-Y. Huang, "Reliability evaluation of service-oriented architecture systems considering fault-tolerance designs", Journal of Applied Mathematics, 2014: 1-11, 2014.

[12] O. Kopp, F. Leymann, and D. Wutke," Fault handling in the web service stack", Service-Oriented Computing, 6470: 303-317, 2010.

[13] A. Carrera, C. A. Iglesias, J. Garcia-Algarra, and D. Kolarik, "A real-life application of multi-agent systems for fault diagnosis in the provision of an internet business service", Journal of Network and Computer Applications, 37: 146-154, 2014.

[14] D. M. Muñoz, A. Correcher, E. García, and F. Morant, "Stochastic DES fault diagnosis with colored interpreted petri nets", Mathematical Problems in Engineering, 2015: 1-13, 2015.

[15] Y. Liu, and L. Qiu, "An approach to web service faults diagnosis based on conditional random fields", International Journal of Control and Automation, 10(2): 129-136, 2017.

[16] X. Han, B. Li, K. F. Wong, and Z. Shi, "Exploiting structural similarity of log files in fault diagnosis for web service composition", CAAI Transactions on Intelligence Technology, 1(1): 61-71, 2016.

[17] G. Fan, A petri net-based byzantine fault diagnosis method for service composition, in 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC), 2012: 42-51.

[18] S. Yin, X. Yang, H. R. Karimi, "Data-driven adaptive observer for fault diagnosis", Mathematical Problems in Engineering, 2012: 1-21, 2012.

[19] Y. Dai, L. Yang, B. Zhang, and Z. Zhu, Exception diagnosis for composite service based on error propagation degree, in 2011 IEEE International Conference on Services Computing (SCC 2011), 2011: 160-167.

[20] S. Duan, H. Zhang, G. Jiang, and X. Meng, Supporting system-wide similarity queries for networked system management, in Proceedings of the 2010 IEEE-IFIP Network Operations and Management Symposium, 2010: 567-574.

[21] P. Kemper, and C. Tepper, "Automated trace analysis of discrete-event system models", IEEE Transactions on Software Engineering, 35(2): 195-208, 2009.

[22] F. Benayas, A. Carrera, and C. A. Iglesias, Towards an autonomic Bayesian fault diagnosis service for SDN environments based on a big data infrastructure, in 2018 Fifth International Conference on Software Defined Systems (SDS), 2018: 7-13.

[23] E. AI-Masri, and Q.H. Mahmoud, Discovering the best web service, in 16th International Conference on World Wide Web (WWW), 2007: 1257-1258.