Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A motifs-based Maximum Entropy Markov Model for realtime reliability prediction in System of Systems

Hongbing Wang[a,*], Huanhuan Fei[a], Qi Yu[b], Wei Zhao[a], Jia Yan[a], Tianjing Hong[a]

[a] *School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing, 211189, China*
[b] *College of Computing and Information Sciences, Rochester Institute of Tech, USA*

ARTICLE INFO

Article history:
Received 21 August 2018
Revised 3 January 2019
Accepted 8 February 2019
Available online 10 February 2019

Keywords:
Service composition
System of Systems
Reliability prediction
Maximum Entropy Markov Model
Time series

ABSTRACT

System of Systems (SoS) based on service composition is considered as an effective way to build large-scale complex software systems. It regards the system as a service and integrates multiple component systems into a new system. The performance of the component system may fluctuate at any time because of the complex and changeable running state and external environment of the component system, which will affect the running of the SoS. The online reliability prediction technology is used to predict the reliability of the component system of an SoS in the near future. It aims to find errors and correct them in time so as to ensure that the SoS can run continuously and smoothly. To tackle the reliability prediction problem of component system in a dynamic and uncertain environment, the paper integrates Maximum Entropy Markov Model (MEMM) with time series motifs to achieve a new prediction model (m_MEMM), which is referred to as motifs-based MEMM. Extensive experiments are conducted to demonstrate the effectiveness and accuracy of the proposed approach.

1. Introduction

As users' requirements on software systems' functionalities have become increasingly complex and diversified, a tightly coupled software development process is difficult to adapt quickly to the changes demanded by the market and consumers. System of Systems (SoS) can effectively solve the above problem as it treats a system as a service and integrates a number of component systems into a new one through service composition. As an emerging way to build large-scale complex softwares, SoS has attracted wide attention in academia and industry (Tekinerdogan and Erata, 2017; Hall et al., 2016; Buscarino et al., 2018).

In an SoS built from service composition, component systems typically run in a complex and dynamic environment. Furthermore, they are distributed and independent of each other. Therefore, careful attention should be given to the collaboration and coordination among component systems in conjunction to the monitoring and quality assurance of the entire SoS. Due to the complexity of the internal running state and external environment of the component systems, their performances may fluctuate at any time, which will affect the performance of the entire SoS. Therefore, quality assurance of the entire SoS is particularly challenging.

Online reliability prediction can help ensure an SoS to continuously run in a healthy state. In particular, it predicts the reliability of the component systems in the near future and the prediction result can be used to prevent and correct mistakes/errors in time. However, due to the complex running environment of the component systems along with their internal running status, it is hard to detect obvious regularities when an error occurs. In general, this prediction problem faces the following challenges:

1. It lacks obvious regularities when the component systems fail. The state change of the system is usually random and uncertain.
2. User activities may lead to volatile behaviors of the system. For example, when a sale promotion is ongoing, there will be a large number of users accessing the system in a short period of time. The system will be under enormous pressure and its performance will be affected.
3. There are limited system parameters that can be used to perform detailed system analysis. Because the component systems are running and maintained separately, it is difficult to obtain hardware layer parameters (such as memory, CPU, network, etc.). On the other hand, application layer parameters, such as throughput and response time, can be conveniently collected through client calls to the component systems.

* Corresponding author.
E-mail addresses: hbw@seu.edu.cn (H. Wang), qi.yu@rit.edu (Q. Yu).

https://doi.org/10.1016/j.jss.2019.02.023
0164-1212/© 2019 Elsevier Inc. All rights reserved.

Table 1
Notations.

SoS	System of Systems
API	Application Program Interface
QoS	Quality of Service
SVM	Support Vector Machine
AMF	Adaptive Matrix Factorization
HMM	Hidden Markov Model
MEMM	Maximum Entropy Markov Model
m_MEMM	motifs-based Maximum Entropy Markov Model
CPD	Conditional Probability Distribution
GIS	Generalized Iterative Scaling
IIS	Improved Iterative Scaling
AVHR	Average Value of Historical Reliability
BR	Bayes' Rules
Reg	Regression

The existing online prediction techniques, such as Bayesian prediction based on conditional probability (Csenki, 1990), non-parametric prediction models (Pfefferman and Cernuschi-Frias, 2002), curve fitting method (Andrzejak and Silva, 2007), semi-Markov models (Salfner, 2006), component interaction graphs (Kiciman and Fox, 2005), SVM (Hoffmann et al., 2006; 2007), and collaborative filtering (Zheng and Lyu, 2010; Yu, 2014), can not fully cope with the complexity of the running environment of an SoS, the uncertainty of the component systems, and the uncertain characteristics of the error events. Most of these online error prediction methods can only model error events that satisfy the Poisson distribution. As for reliability time series prediction problems of uncertain error events, these methods still lack sufficient support.

The paper proposes an online reliability prediction method based on a probability graph model—Maximum Entropy Markov Model (MEMM). The proposed prediction model, referred to as m_MEMM, integrates MEMM with time series motifs. The model will be trained by the historical system parameters of the component systems, and then the reliability prediction of the component system can be achieved.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 presents some preliminaries about prediction model, including Hidden Markov Model, Maximum Entropy Model, and Maximum Entropy Markov Model. Section 4 introduces the definition of reliability and the concept of time series. It also formulates the research problem. In Section 5, the motifs-based Maximum Entropy Model is proposed for reliability prediction. In Section 6, extensive experiments are conducted to verify the correctness and accuracy of the prediction model. Section 7 concludes the paper and identifies some important future work. Table 1 shows the major notations used in this paper.

2. Related work

Reliability prediction has received significant attention from multiple communities, including software engineering and service computing. Zheng and Lyu (2010) developed a reliability prediction method for service systems. Collaborative filtering was adopted to achieve the prediction purpose by using the historical user-service interactions. Specifically, performance parameters of services were collected by calling services. The probability that the service calls fail is recorded. Finally, reliability is calculated as the average error probability of the same service with multiple users over a period of time. In this paper, we follow a similar approach to collect performance parameters. Zhu et al. (2014) proposed a QoS prediction method based on Adaptive Matrix Factorization (AMF) in order to make adaptive decisions in a timely and accurate manner and predict effectively the QoS values for component services.

The Bayesian predicting method (Csenki, 1990) is simple and practical because it predicts the error probability of future mistakes through historical mistakes. However, it only supports error events that follow Poisson distribution, making it difficult to deal with dynamic and uncertain error events. The semi-Markov model-based prediction method (Vaidyanathan and Trivedi, 1999) estimates the system resource consumption rate through the load, and then predicts the system's future errors. This method only considers the impact of the load on the system errors, and ignores other potential factors for errors. The prediction method based on regression analysis (Andrzejak and Silva, 2007) is to use the historical reliability data of the system to fit the reliability trend, and then perform the prediction of future system reliability. However, this curve fitting method is difficult to deal with the prediction of random and uncertain error events.

Amin et al. (2012b, 2012a, 2013) carried out system reliability prediction from the perspective of time series, and proposed a statistical time series model. It is a forward prediction where the prediction of the current step relies on the result of the previous step. This may lead to cumulative errors so that it is difficult to cope with the online reliability time series predictions studied by this paper. Cheung (2008) used the Markov chain to model the dynamic behaviors of traditional software components. They analyzed the execution process of the components, calculated the probability of behavior transition, and then obtained the reliability of software components. This prediction method does not analyze component behavioral characteristics, leading to a poor prediction accuracy. Silic et al. (2013, 2014, 2015) proposed a clustering-based prediction method that used data about users, services, and environment to find services similar to the one to be predicted. They considered these services' reliability as the ultimate prediction result. This method is based on the historical reliability of other services, which may not lead to accurate prediction result.

Since the hardware layer parameters of the component systems are difficult to obtain, this paper collects application layer parameters by invoking the component systems. These parameters are then used to build a prediction model that combines the Maximum Entropy Markov Model with time series motifs. The proposed model is able to deal with the dynamic and uncertain environment of component systems and make accurate online reliability prediction as demonstrated through our experiments.

3. Preliminaries

In this section, we introduce the Maximum Entropy Markov Model, which is a probabilistic graph model for time-series variables. It combines the advantages of Hidden Markov Model and Maximum Entropy Model, both of which will be briefly described.

3.1. Hidden Markov Model

The Hidden Markov Model (HMM) (Baum et al., 1970; Baum and Petrie, 1966) is an extension of the Markov process. While its states cannot be directly observed, their distribution can be inferred from observations. An HMM is a finite model that describes the probability distribution between states and observations. It is commonly used in many applications including natural language processing and speech recognition.

Definition 1 (Hidden Markov Model)**.** A Hidden Markov Model can be defined as a triplet $HMM = < P, A, B >$, where

- P represents the probability distribution of the initial state;
- A is the state transition matrix, $A = [a_{ij}]$, where a_{ij} is the probability of transition from state a_i to state a_j;

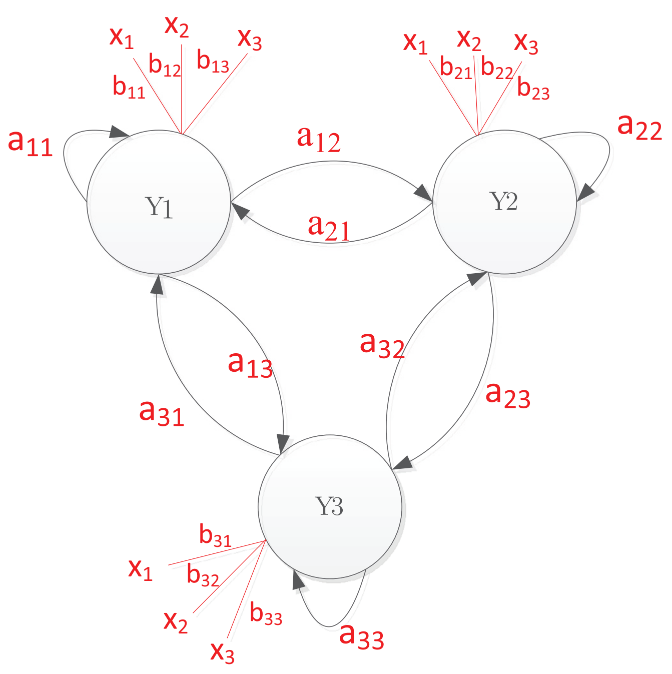


Fig. 1. Hidden Markov Model.

- B is the observation distribution matrix, $B = [b_{ik}]$, where b_{ik} means that for the a_i state, the probability of the k observation is b_{ik}.

A typical HMM is shown in Fig. 1. It has three hidden states Y_1, Y_2, and Y_3 and their transition probabilities are given by a_{ij}. Each hidden state has three visible states: X_1, X_2, and X_3, and they are observed by the probability b_{ik}. Both matrices A and B can be learned from sample data. In fact, an HMM can be established based on two basic assumptions:

- **First-order Markov assumption**: The current state is only related to the previous state and has nothing to do with the earlier state.
- **Observation independence assumption**: The current observations are only related to the current state and have nothing to do with other states.

Many real-world applications do not meet these two assumptions, leading to poor modeling accuracy of an HMM. More discussions will be given in Section 3.3.

3.2. Maximum Entropy Model

Before introducing the Maximum Entropy Model, we first provide a definition of entropy.

3.2.1. Concept of entropy

Information entropy is used to represent the measure of uncertainty. The greater the uncertainty, the greater the entropy.

Definition 2 (Entropy)**.** Given a random variable X, its probability distribution is $P(x)$, and its entropy is defined as:

$$H(X) = -\Sigma_x P(x) log P(x) \tag{1}$$

For the two random variables X and Y, their joint entropy is defined as:

$$H(X, Y) = -\Sigma P(x, y) log P(x, y) \tag{2}$$

In the formula 2, $P(x, y)$ is the joint probability distribution of random variables X and Y. According to the joint entropy, the conditional entropy for X and Y is defined as follows:

$$H(Y|X) = H(X, Y) - H(X) = -\Sigma P(x, y) log P(y|x) \tag{3}$$

3.2.2. Mathematical model of maximum entropy

Assume there is a classification model $P(Y|X)$, where X represents the input data and Y represents the output data. There is a large amount of sample data $D = \{(x_i, y_i)|1 \le i \le n\}$, and the empirical joint probability distribution $\bar{P}(x, y)$ of X and Y and the empirical distribution $\bar{P}(x)$ of X can be computed easily.

$$\bar{P}(x, y) = \frac{number(x, y)}{n} \tag{4}$$

$$\bar{P}(x) = \frac{number(x)}{n} \tag{5}$$

The $number(x, y)$ and $number(x)$ in the formulas 4 and 5 indicate the total number of occurrences of the samples (x, y) and (x) in the sample set D, respectively.

In the Maximum Entropy Model, there is a two-valued feature function describing a kind of association of x and y, or certain rules. Its definition is as follows:

$$f(x, y) = \begin{cases} 1 & x \text{ and } y \text{ have some correlations} \\ 0 & \text{Others} \end{cases} \tag{6}$$

The empirical expectation of the feature function is $E_{\bar{p}}(f)$, and the true expectation of the feature function is $E_p(f)$. They are defined as follows:

$$E_{\bar{p}}(f) = \sum \bar{P}(x, y) f(x, y) \tag{7}$$

$$E_p(f) = \sum \bar{P}(x) P(y|x) f(x, y) \tag{8}$$

The Maximum Entropy Model can be learned from the training set. We need to make an assumption on these two expectations (see formula 9). If the total number of feature functions is m, there are m constraints.

$$E_{\bar{p}}(f_a) = E_p(f_a)(a = 1, 2, \ldots, m) \tag{9}$$

The conditional entropy of the Maximum Entropy Model $P(Y|X)$ is:

$$H(P) = -\sum \bar{P}(x) P(y|x) \log P(y|x) \tag{10}$$

The goal of the entire model is to calculate the conditional probability $P(y|x)$ when the conditional entropy $H(P)$ is maximum under the constraint condition (formula 9).

The Maximum Entropy Model problem can be transformed into an optimization problem with constraints, which can be solved using the Lagrangian multiplier method. The result is given by formula 11.

$$P_\omega(y|x) = \frac{exp(\sum_{a=1}^{m} \omega_a f_a(x, y))}{N_\omega(x)}$$
$$N_\omega(x) = \sum_y exp\left(\sum_{a=1}^{m} \omega_a f_a(x, y)\right) \tag{11}$$

In formula 11, ω_a is the weight of the ath feature function, and $N_\omega(x)$ is the Normalization Function.

The Maximum Entropy Model is a model with maximum entropy under all constraints, and the uncertain part of the model is set to equal probability (ie, the maximum entropy), thereby reducing the risk. Because there is no independence assumption like HMM, the selection of features in the Maximum Entropy Model will not be restricted. The complex and related features can be flexibly selected as constraints, and the fitness of the model to the unknown data can be adjusted by the number of constraints.

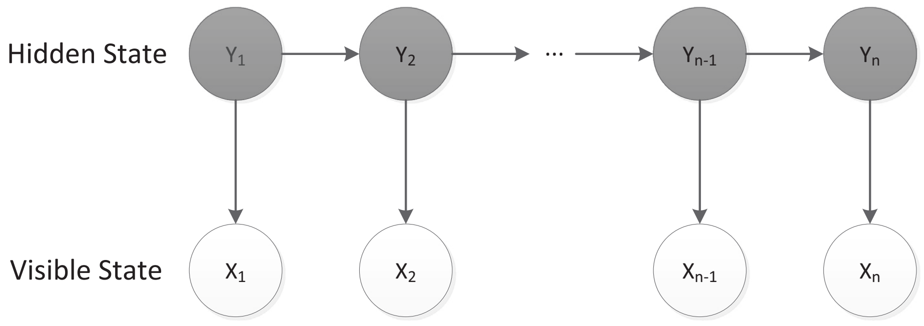


Fig. 2. Hidden Markov Model.

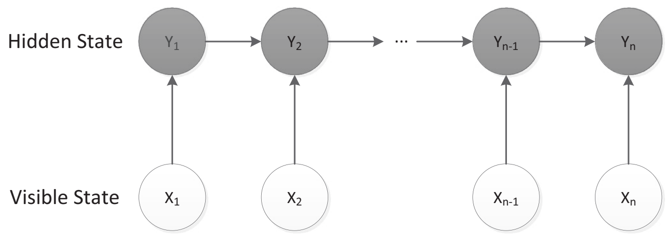


Fig. 3. Maximum Entropy Markov Model.

3.3. Maximum Entropy Markov Model

The Maximum Entropy Markov Model (MEMM) is an extension of an HMM where the Maximum Entropy Model is introduced to combine the advantages of the two models. In the MEMM, a probability distribution $P(y|y', x)$ is used to replace the two probability distributions in the Hidden Markov Model, namely the state transition probability distribution $P(y|y')$ and the observation probability distribution $P(x|y)$. Where $P(y|y', x)$ means the probability that the current state is y under the conditions that the previous state is y' and the observed value is x.

Figs. 2 and 3 show HMM and MEMM, respectively. Their key difference lies in the direction between the hidden states and the visible states. In the HMM, because of its observation independence assumption that the observation of the current state is only related to the current state and has nothing to do with the other states, it can be seen from the figure that each observed value of x only comes from a state y. In addition, the two arrows in Fig. 2 represent two conditional probabilities, namely, the state transition probability and the observation probability. However, these two conditional probabilities are not the final objective function. The objective function needs to be transformed into a posterior probability through the Bayesian formula. In the MEMM, only one conditional probability distribution $P(y|y', x)$ is needed to describe the entire model, and the conditional probability distribution is also the final objective function. It can be seen from the arrow that the current observation and the previous state jointly determine the current state. Conditional probability $P(y|y', x)$ referred to as $P_{y'}(y|x)$ in the following discussion.

HMM follows the Markov property to be able to describe the transition relationship among different states. Due to the observation independence assumption, it is not possible to select features flexibly. On the contrary, the Maximum Entropy Model can flexibly select features and solve the optimal conditional probability $P(y|x)$ based on the known empirical distribution. However, it cannot capture the relationship among states due to lack of Markov chain like HMM. Therefore, the MEMM combines an HMM and a Maximum Entropy Model to form a unified generative model to take full advantage of the two models and make up for each other's shortcomings.

An MEMM uses a state-observation transition function to replace the separated state and observation functions in an HMM. This state-observation transition function can model observations and the transitions between states. Like the Maximum Entropy Model, the definition of the feature function in the MEMM is unchanged (see the formula 6), but its two parameters have a specific meaning: x represents the observed value, and y represents the current state.

Consider an observation sequence $(x_1, x_2, \dots, x_n)$ and the corresponding sequence of states $(y_1, y_2, \dots, y_n)$. The empirical expectation of a feature function is $E_{\tilde{p}}(f)$, which is calculated as:

$$E_{\tilde{p}}(f) = \frac{1}{n}\sum_{i=1}^{n} f(x_i, y_i) \tag{12}$$

We can calculate the true expectation $E_p(f)$ of the feature function using the estimated conditional probability:

$$E_p(f) = \frac{1}{n}\sum_{i=1}^{n}\sum_{y} P_{y'}(y|x_i) f(x_i, y) \tag{13}$$

The Maximum Entropy Model considers the two expectations to be equal, as given by the formula 14,

$$E_{\tilde{p}}(f_a) = E_p(f_a)(a = 1, 2, \dots, m) \tag{14}$$

where m is the total number of feature functions. Finally, an empirical distribution $P_{y_{i-1}}(y_i|x_i)$ can be available, which means the probability that the current state is y_i when the previous state of the current state is y_{i-1}, and the current observation is x_i.

$$P_{y_{i-1}}(y_i|x_i) = \frac{exp(\sum_{a=1}^{m} \omega_a f_a(x_i, y_i))}{N_\omega(x_i, y_{i-1})}$$
$$N_\omega(x_i, y_{i-1}) = \sum_{y} exp\left(\sum_{a=1}^{m} \omega_a f_a(x_i, y_i)\right) \tag{15}$$

where $i = 1, 2, \dots, n$, ω_a is the weight of the ath feature function, and $N_\omega(x_i, y_{i-1})$ is a normalization function. Note that each feature function has a weight which is determined based on the maximum entropy principle, so that we can get the optimal conditional probability $P_{y_{i-1}}(y_i|x_i)$ under the constraints.

4. Problem formulation

Due to the randomness and uncertainty of system errors, we consider that the running status of the system in the near future is only related to the status of the latest period of time, and has nothing to do with the earlier running status. Therefore, we need an online prediction technology to predict the reliability of the system in the future by observing the current system running status (current application layer parameters of the system) with the trained model.

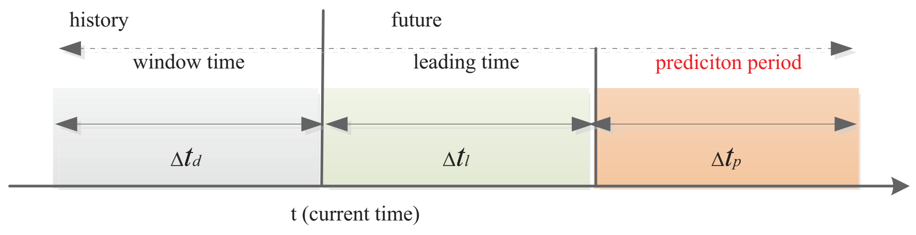


Fig. 4. Prediction period.

As shown in Fig. 4, Δt_d is the window time, which corresponds to the historical running status of the component systems. We can use this period of time to train the model and learn its parameters. Δt_l is the leading time, which is the time when the client recently invoked the component systems. It starts at the current time t and ends at the end of the call. Δt_p is a valid prediction period, which starts at the time point when the leading time ends. Our goal is to predict the reliability of the component system in the near future (Δt_p) based on the behaviors of the component systems during the recent time (Δt_l) using a prediction model which is trained by historical data in Δt_d.

More specifically, we are looking for a function f such that

$$\vec{P_r} = f(\vec{P_{r'}}, \vec{D}_{rt}, \vec{D}_t) \tag{16}$$

where $\vec{P_r}$ is the reliability time series of the component system in the near future t, $\vec{P_{r'}}$ is the reliability time series of the component system at current t', $\vec{D}_{rt}$ is the response time series of the client-invoked component systems, and $\vec{D}_t$ is the throughput series of component systems. The time t and t' are two adjacent time segments. It is worth to note that the reliability $\vec{P_r}$ of the component system to be predicted is not for a specific point but a time period. Therefore, it is a time series prediction problem.

The concept of reliability may be interpreted differently in diverse disciplines. For software engineering, the definition of reliability given by the ISO/IEC/IEEE International Standard (ISO/IEC/IEEE 24765:2010, 2010) consists of two descriptions: (1) Reliability is the ability of a system or component to perform specified functions for a period of time under specified conditions; (2) The ability of a software product to maintain a certain level of performance under certain conditions. This definition is widely used in the measurement of the reliability of software systems, and the same applies to component systems of service-based SoS. However, this definition only gives a qualitative description of reliability, and a quantitative one is needed for our purpose.

Software system reliability metrics are measured in many forms, including (1) failure rate, (2) mean time between failures, (3) mean time to failure, (4) demand failure rate. The component system of the SoS based on the service composition leads to the inconspicuous feature of the component system failure due to the dynamics and uncertainties of the internal state of the system, the external environment, and the load of the component system. Therefore, the failure rate, mean time between failures, and mean time to failure are not suitable for describing the reliability of such a dynamic and uncertain component system. Demand failure rate can be used to describe the reliability of a component system (Brosch et al., 2012; Wang et al., 2017), which means that the system does not have a failure at time t, and the probability of system running failure after t. In this paper, we consider that the demand fails in two cases. First, the system gives a wrong response. For example, due to a problem with the internal state of the system, it gives an error response, or no response. Second, system response time is too long. For example, due to the external network environment of the system, the system response times out.

According to Brosch et al. (2012) and Wang et al. (2017), we give a formal definition of the reliability of the component system:

Definition 3 (Component System Reliability)**.** Let δ be a certain period, and t_δ be any point in the period δ. The component system is called multiple times during this period, and the number of failed calls is N_{error}. The number of failed calls per unit time is denoted as μ, and the reliability is defined as:

$$\begin{aligned} r(\delta) &= e^{-\mu} \\ \mu &= \frac{N_{error}}{\delta} \end{aligned} \tag{17}$$

Note that the definition of the reliability of the component system is described by an exponential function. When the number of failed calls per unit time is $mu = 0$, the reliability is $r(\delta) = e^{-\mu} = e^{-0} = 1$ indicates that the component system is running normally and reliable during this time. When the number of failed calls per unit time is $mu \to \infty$,

$$r(\delta) = \lim_{mu \to \infty} e^{-\mu} = 0$$

indicates that some kind of fault or error occurs in the operation within the time for the component system, leading to a sharp decline in reliability.

Based on Definition 3, let's consider a certain period ΔT for a component system. We divide ΔT into n contiguously equal time slices, $\delta_1, \delta_2, \ldots, \delta_n$. For each time slice to call the component system, the reliability on each time slice is calculated according to the formula 17, and finally the reliability time series on ΔT is obtained, $p_{r_1}, p_{r_2}, \ldots, p_{r_n}$. As shown in Fig. 5, $\delta_i = 200$ ms, $\sum_{i=1}^{n} \delta_i = 2s$, the figure reflects the reliability time series of a component system in 2s, $\vec{P_r} = (0.9, 0.8, 1, 0.8, 0.7, 0.9, 1, 0.8, 0.9, 1)$.

A service-based SoS integrates multiple existing component systems into a larger system through service composition to meet more complex user requirements. Each component system encapsulates the functions into a service, such as a SOAP-based service or a RESTful-based service. Through the service composition technology, numerous services are integrated to form a fully functional and powerful SoS. In order to ensure the overall running of the SoS, it is necessary to ensure that each component system (service) can run reliably. This type of reliability is an online one which describes the running status of component systems in the near future. Therefore, this paper needs to predict this online reliability of the component system.

Because different users have different lengths of call time for the component system, the requirements for the continuous and stable running of the component system are not the same. In order to ensure that the predicted results can adapt to different application requirements, this article predicts the online reliability time series of component systems. The following is the definition of online reliability time series:

Definition 4 (Online Reliability Time Series)**.** Let Δt_p be the effective prediction period (see the Fig. 4), divide Δt_p into n equal continuous time slices, namely $\delta_1, \delta_2, \ldots, \delta_n$. We define the online reliability time series $(\vec{P_r})$ of the component system as a n dimensional vector, i.e., $\vec{P_r} = (p_{r_1}, p_{r_2}, \ldots, p_{r_n})$ Where p_{r_i} is the reliability of the component system during the slice δ_i, i.e., $p_{r_i} = r(\delta_i)$.

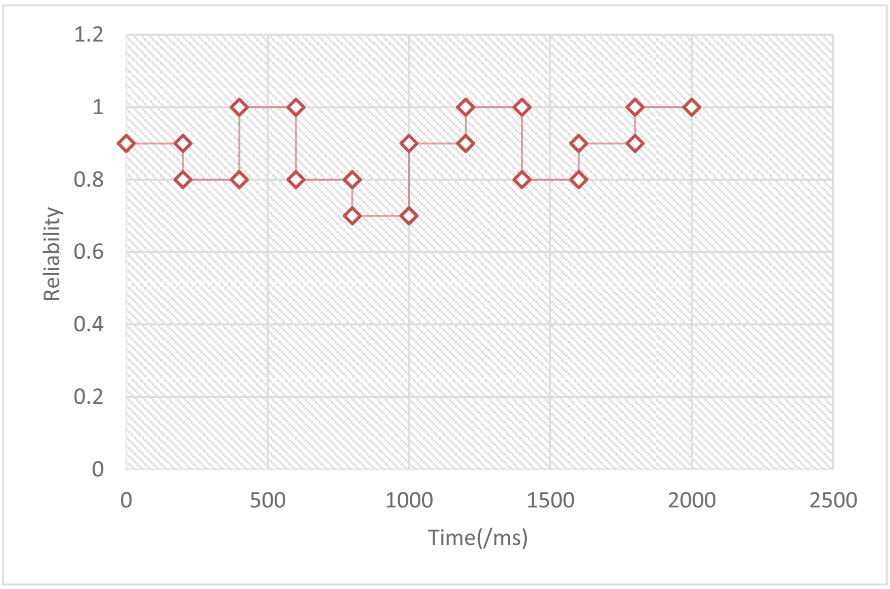


Fig. 5. Reliability time series.

In general, the same component system has different running states at different times, showing different reliability time series. When the same component system is invoked by different clients, it may deliver different performance. As a result, its online reliability time series will be different. Therefore, this paper is concerned with the reliability prediction with regard to a specified client for a given component system.

5. Prediction method

In this section, we describe in detail the motifs-based Maximum Entropy Markov Model (m_MEMM) for online reliability time series prediction of component systems. The system parameters of the component system including the response time, throughput and historical reliability are obtained through client-side service invocations. As shown in Fig. 6, we use time series motifs to represent each node in the m_MEMM. In the figure, $\widehat{D}_{rt}$ and $\widehat{D}_t$ are the response time series motifs and the throughput time series motifs, respectively, which are observations that describe the visible state in the MEMM. $\widehat{P}_r$ is the reliability time series motifs used to describe the hidden state in MEMM. The reliability time series motifs describe the running state of a component system, and this running state is not observable, so we consider it is reasonable to use it to describe the hidden state in MEMM.

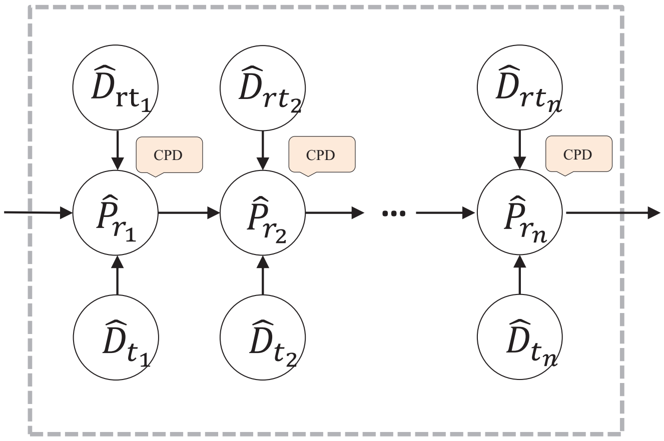


Fig. 6. Motifs-based Maximum Entropy Markov Model.

The time series motifs refer to the feature values in the historical system parameters of the component system. Specifically, considering the long-term historical system parameters of the component system, this period can be divided into a number of consecutive equal-length time slices, so that multiple time series of historical parameters of component system can be obtained. Feature values can be found through unsupervised learning (e.g, clustering), which are referred to as *time series motifs*. The formal definition is as follows:

Definition 5 (Time Series Motifs)**.** Let D be a long-term system parameter of the component system. Divide this system parameter into n equal-length continuous time series, denoted as $\vec{D} = (\vec{D}_1, \vec{D}_2, \ldots, \vec{D}_n)$. By clustering the time series of multiple system parameters, we get k cluster center points. These center points are called time series motifs on D and are denoted as $\widehat{D} = (\widehat{D}_1, \widehat{D}_2, \ldots, \widehat{D}_k)$.

Fig. 7 shows a basic unit of a MEMM model, where the red part indicates the reliability time series motifs of the component systems within the effective prediction time Δt_p. $\widehat{P}_{r'}$ represents the reliability time series motifs for the previous state, $\widehat{D}_{rt}$ and $\widehat{D}_t$ represents the time series motifs of the observation parameters of the recent time period Δt_l. As indicated by the arrow in the figure, $\widehat{P}_{r'}$, $\widehat{D}_{rt}$ and $\widehat{D}_t$ jointly determine the Conditional Probability Distribution (CPD) of the time series motifs in the effective prediction period. Below is the definition of m_MEMM:

Definition 6 (m_MEMM)**.** A motifs-based Maximum Entropy Markov Model (m_MEMM) can be defined as a four-tuple $< H, H'$, $O, CPD >$, where each node in the model is the time series motifs of the corresponding parameters. More specifically,

- H is the current state node.
- H' is the previous node of the current state node.
- O is the set of observations in the current state, that is, the collection of system parameters time series motifs.
- CPD is the Conditional Probability Distribution (CPD) satisfying maximum entropy, which describes the probability $P(H|H', O)$ where the current state is H when the observation in the current state is O, and the previous state is H'.

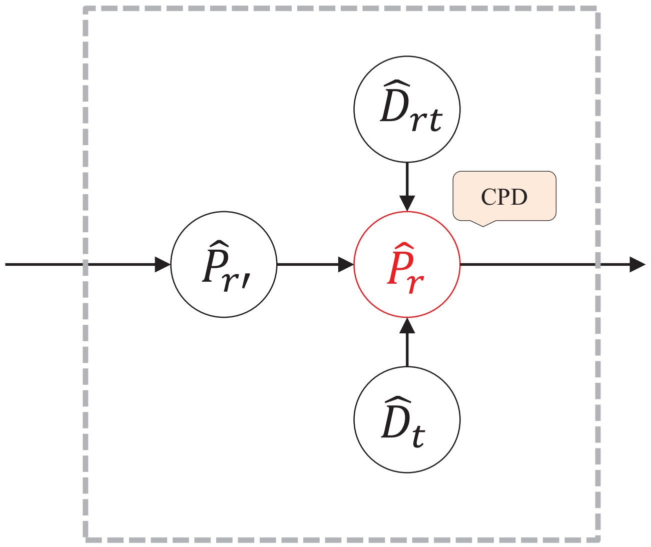


Fig. 7. A base unit of m_MMEM.

The m_MEMM model combines the advantages of HMM and Maximum Entropy Model, which allows it to capture the transition relationship among hidden states while using a conditional probability distribution to meet the maximum entropy.

The final exponential distribution in MEMM is $P_{y_{i-1}}(y_i|x_i)$ (see the formula 15). After introducing motifs, the variables x and y have new meanings. The variable x represents the response time series motifs $\widehat{D}_{rt}$ and the throughput time series motifs $\widehat{D}_t$, and the variable y represents the reliability time series motifs $\widehat{P}_r$. Then, the expression for each conditional probability distribution CPD is:

$$P(\widehat{P}_{r_i} \,|\widehat{P}_{r_{i-1}}, \widehat{D}_{rt_i}, \ \widehat{D}_{t_i}) = \frac{exp(\sum_{a=1}^{m} \omega_a f_a(\widehat{D}_{rt_i}, \widehat{D}_{t_i}, \ \widehat{P}_{r_i}))}{N_\omega(\widehat{D}_{rt_i}, \widehat{D}_{t_i} \ \widehat{P}_{r_{i-1}})}$$
$$N_\omega(\widehat{D}_{rt_i}, \widehat{D}_{t_i} \ \widehat{P}_{r_{i-1}}) = \sum_{\widehat{P}_{r_i}} exp\left(\sum_{a=1}^{m} \omega_a f_a(\widehat{D}_{rt_i}, \widehat{D}_{t_i}, \ \widehat{P}_{r_i})\right) \tag{18}$$

Based on the m_MEMM model as described above, online reliability prediction of component systems can be achieved through the following steps:

1. Data collection and preprocessing, which divides the historical system parameters of the collected component system into equal time series.
2. The time series motifs computation, which takes the time series of the well-divided component system parameters as input, and searches for the time series motifs of historical system parameters through clustering.
3. Marking time series motifs, which collects component system parameters of the leading time in real time, and calculates the nearest time series motifs.
4. Parameter estimation, which estimates the motifs-based Maximum Entropy Markov Model parameters through the training of the historical system parameters of the component system.
5. Reliability prediction, which predicts the reliability of the component system in the near future.

5.1. Data collection and preprocessing

In this step, the collected system parameters need to be preprocessed so that they can be used for model training. Historical system parameter data of the component system can be collected through client-side invocations of the component system. These

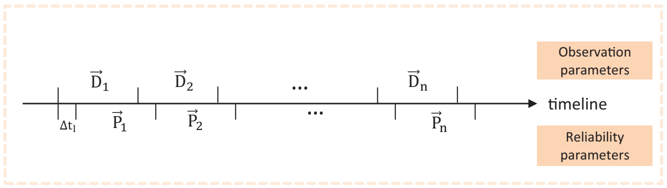


Fig. 8. A schematic diagram of time series division of historical component system parameters.

parameters include response time, throughput, and historical reliability. We divide long-term historical system parameters of the component system into multiple equal-length contiguous time series, as shown in Fig. 8.

We divide the historical parameters of the component system into two types: one is the observation parameter, which means that the component system parameters can be directly obtained, including the response time and throughput parameters. They correspond to the visible state in the m_MEMM. The other type is the reliability parameter, which indicates the reliability of the component system. It describes the running status of the component system and is the hidden state in the m_MEMM. The division of these two types of parameters differs by one leading time Δt_l (see Fig. 4).

According to the division method shown in Fig. 8, we will get the response time series $\vec{D}_{rt}$, the throughput time series $\vec{D}_t$, and the reliability time series $\vec{P}_r$. According to the definition of m_MEMM (see Definition 6), a sample for training the model should contain four sets of data: response time series $\vec{D}_{rt}(i)$, throughput time series $\vec{D}_t(i)$, reliability time series of current time $\vec{P}_r(i)$ and reliability time series of previous time $\vec{P}_r(i-1)$, so it is denoted as $sample = (\vec{D}_{rt}(i), \vec{D}_t(i), \vec{P}_r(i), \vec{P}_r(i-1))$.

5.2. Time series motifs computation

The component system parameter time series motifs represent the feature values of the system parameter time series. Motifs are found by clustering a large number of system parameter time series. The center points of the clustering are called as the time series motifs. The m_MEMM model uses the time series motifs to represent each node of the model. For this purpose, the sample data obtained in the first step needs to be clustered separately. We need cluster the time series $\vec{D}_{rt}, \ \vec{D}_t, \ \vec{P}_r$.

In this paper, we use the K-Means algorithm for time series clustering. K-Means is a classical clustering algorithm with good efficiency and scalability. In order to measure the similarity between data, Euclidean distance is used to calculate the distance between two time series (see the formula 19).

$$\|\vec{D}_1, \vec{D}_2\| = \sqrt{\sum_{i=1}^{n} (\vec{D}_1(i) - \vec{D}_2(i))^2} \tag{19}$$

where $\vec{D}_1$ and $\vec{D}_2$ represent two different system parameter time series, $\vec{D}_1(i)$ represents the i-th dimension of time series $\vec{D}_1$, and n represents the total dimension.

Here we take the reliability time series of the component system $\vec{P}_r$ as an example. After running the K-Means clustering algorithm, we will get k clusters (center points), which are denoted as:

$$\widehat{P}_r = (\widehat{P}_{r_1}, \widehat{P}_{r_2}, \ldots, \widehat{P}_{r_k}) \tag{20}$$

5.3. Marking time series motifs

At this step, we need to mark the time series of the system parameters. Marking refers to computing a distance between $\vec{D}$ and

k time series motifs. We mark it with the nearest motif to the time series. It can be denoted as:

$$\vec{D} \to arg\min_{\widehat{D}_i}\{\|\vec{D}, \widehat{D}_i\|\} \qquad i = 1, 2, \ldots, k \tag{21}$$

The time series motifs reflect the characteristics of the time series of the component system. The time series of each time slice can be marked by the nearest time series motifs. For example, the following two time series motifs, $\widehat{D}_1$ and $\widehat{D}_2$ are obtained in the sample data:

$$\begin{aligned}\widehat{D}_1 &= (0.72,\ 0.85)\\ \widehat{D}_2 &= (0.63,\ 0.88)\end{aligned} \tag{22}$$

There is a time series $\vec{D} = (0.66, 0.78)$, and we calculate the Euclidean distance from $\widehat{D}_1$ and $\widehat{D}_2$ as:

$$\begin{aligned}\|\vec{D}, \widehat{D}_1\| &= \sqrt{(0.66-0.72)^2 + (0.78-0.85)^2} = 0.0922\\ \|\vec{D}, \widehat{D}_2\| &= \sqrt{(0.66-0.63)^2 + (0.78-0.88)^2} = 0.1044\end{aligned} \tag{23}$$

From the calculation of the formula 23, it can be concluded that the time series $\vec{D}$ is nearer to $\widehat{D}_1$, then $\vec{D}$ is marked as $\widehat{D}_1$, which is written as $\vec{D} \to \widehat{D}_1$.

By marking the time series divided in the first step, we will get the response time series motifs $\widehat{D}_{rt}$, the throughput time series motifs $\widehat{D}_t$, and reliability time series motifs $\widehat{P}_r$.

5.4. Parameter estimation

At this step, we will train the m_MEMM model through a large number of component system parameter data and estimate the parameters of the model. Specifically, we need to calculate conditional probability distribution $P(\widehat{P}_{r_i}|\widehat{P}_{r_{i-1}}, \widehat{D}_{rt_i}, \widehat{D}_{t_i})$ (See the formula 18), i.e., the weight of the feature function ω_a, whose value is determined according to the principle of maximum entropy.

In the reliability prediction of the component system, we need to redefine the feature function f (see the formula 24).

$$f_i(\widehat{D}_{rt_i}, \widehat{D}_{t_i},\ \widehat{P}_{r_i}) = \begin{cases}1 & <\widehat{D}_{rt_i}, \widehat{P}_{r_i}> \text{ or } <\widehat{D}_{t_i}, \widehat{P}_{r_i}>\\ 0 & \text{Others}\end{cases} \tag{24}$$

where $<\widehat{D}_{rt_i}, \widehat{P}_{r_i}>$ means the time series motifs $\widehat{D}_{rt_i}$ and the time series motifs $\widehat{P}_{r_i}$ appear together, or in a sample. In particular, note that the two motifs have the same index i. $<\widehat{D}_{t_i}, \widehat{P}_{r_i}>$ has the same meaning as above.

Suppose we record the number of corresponding response time series motifs, throughput time series motifs, and reliability time series motifs as k, and the number of feature functions $f_a(\cdot)$ as M. According to the definition of the feature function it can be concluded that $M = k$.

Note that $\widehat{D}_{rt_i}$ and $\widehat{D}_{t_i}$ are observation parameter data (see Fig. 8), describing the visible parameters of the component system; The $\widehat{P}_{r_i}$ belongs to the reliability parameter data and describes the running status of the component system. The definition of this feature function clearly reflects the causal relationship between the two sets of parameters. In the following text, for ease of writing, feature function $f_a(\widehat{D}_{rt_i}, \widehat{D}_{t_i},\ \widehat{P}_{r_i})$ is written as $f_a(\cdot)$.

The Maximum Entropy Markov Model is essentially an extension of the Maximum Entropy Model. The commonly used algorithms for solving the Maximum Entropy Model are the generalized iterative algorithm (Darroch and Ratcliff, 1972) (Generalized Iterative Scaling, GIS) and the improved iteration scale algorithm (Berger et al., 1996) (Improved Iterative Scaling, IIS). Because GIS algorithm needs to iterate many times to converge and has low efficiency, this paper uses IIS algorithm to solve the model m_MEMM proposed in this paper. The specific solution steps mainly include the following steps:

1. calculate the empirical expectation of each feature function

$$F_a = \frac{1}{n}\sum_{i=1}^{n} f_a(\cdot) \tag{25}$$

2. compute the true expectation $E_a^{(j)}$ for each feature function, where j represents the jth iteration.

$$E_a^{(j)} = \frac{1}{n}\sum_{i=1}^{n}\sum_{y} P_{y'}(y|x) f_a(\cdot) \tag{26}$$

Among them, x represents the observation data, namely the response time series motifs and the throughput time series motifs, and y represents the reliability parameter, i.e., the reliability time series motifs.

3. solve the equation about $\Delta\omega_a$:

$$\sum_{x,y}\bar{p}(x)p_{y'}(y|x)F_a(\cdot)exp(\Delta\omega_a\sum_{a=1}^{M} f_a(\cdot)) = F_a \tag{27}$$

Since $\sum_{a=1}^{M} f_a(\cdot)$ is not a constant, we use Newton's method to solve and let $g(\omega_a) = 0$ denoted as Eq. (27).

$$\Delta\omega_a^{(n+1)} = \Delta\omega_a^{(n)} - \frac{g(\Delta\omega_a^{(n)})}{g'(\Delta\omega_a^{(n)})} \tag{28}$$

4. repeat steps 2 and 3. When $|\Delta\omega_a^{(n+1)} - \Delta\omega_a^{(n)}|$ is small enough, the algorithm is considered as converged. For a detailed algorithm description, the Algorithm 1 shows all steps.

Algorithm 1: m_MEMM model training.

```
1 input: component system history parameters: response time
  series motifs $\widehat{D}_{rt}$, throughput time series motifs $\widehat{D}_t$, and
  component system history reliability time series motifs $\widehat{P}_r$
2 output: m_MEMM model parameter $\omega_a$

1:  Set $\omega_a^{(0)}$ to an initial value of 1
2:  repeat
3:    for each feature function do
4:      compute empirical expectation of feature function
        $F_a = \frac{1}{n}\sum_{i=1}^{n} f_a(\cdot)$
5:      repeat
6:        for each iteration do
7:          In the $j$-th iteration, calculate the true expectation
            of each feature function using the current weight
            $\Delta\omega_a^{(j)}, E_a^{(j)} = \frac{1}{n}\sum_{i=1}^{n}\sum_{\widehat{P}_r} P_{\widehat{P}_{r'}}(\widehat{P}_r|\widehat{D}_{rt}, \widehat{D}_t)F_a(\cdot)$
8:          Solve the equation about $\Delta\omega_a$ (see formula 27),
            According to the formula 28, the weight of the
            $(n+1)$-th iteration can be obtained: $\Delta\omega_a^{(n+1)}$
            $\Delta\omega_a^{(n+1)} = \Delta\omega_a^{(n)} - \frac{g(\Delta\omega_a^{(n)})}{g'(\Delta\omega_a^{(n)})}$
9:        end for
10:     until The difference between $\Delta\omega_a^{(n)}$ and $\Delta\omega_a^{(n+1)}$ is
        small enough
11:   end for
12: until All feature function weights are solved
```

5.5. Reliability prediction

At this step, we will collect the relevant parameter data of the component systems of the recent time and predict the reliability in future.

1. Collect observations, including the current time component system reliability time series $\vec{P}_{r'}$, system's response time series $\vec{D}_{rt}$

and the throughput time series $\vec{D}_t$ in adjacent time period Δt_l (see Fig. 4). Where $\vec{P}_{r'}$ is obtained by invoking the component system and referring to the Definition 3.

2. Mark time series motifs, referring to the Section 5.3, the time series $\vec{P}_{r'}$, $\vec{D}_{rt}$ and $\vec{D}_t$ for time series motifs will be marked. They are denoted as:
   - $\vec{P}_{r'} \rightarrow \widehat{P_{r'}}$
   - $\vec{D}_{rt} \rightarrow \widehat{D}_{rt}$
   - $\vec{D}_t \rightarrow \widehat{D}_t$
3. Calculate the conditional probability. Specifically, for each reliability time series motif $\widehat{P}_{r_i}$, we substitute the time series motifs obtained in second step into the formula 24, compute feature function values, and then take the calculation result into the formula 18 to calculate the conditional probability $P(\widehat{P}_{r_i} | \widehat{P}_{r_{i-1}}, \widehat{D}_{rt_i}, \widehat{D}_{t_i})$. Where $\widehat{P}_{r_i}$ is a certain reliability time series within the valid prediction period of the component system in the time series motifs finding step (see Section 5.2).
4. Analyse the prediction result. Comparing the calculation result in the third step, we use the reliability time series motif with the largest condition probability $\widehat{P}_{r_\theta}$ as the final predicted online reliability time series ($\vec{P}_r$), i.e., $\vec{P}_r = \widehat{P}_{r_\theta}$.

6. Experiment and analysis

In this section, we conduct a series of experiments to assess the proposed prediction model and method. We first report the prediction result and compare it with other competitive models. We then study the impact of different factors, including the number of motifs, the size of the dataset, and the length of the time series. At last, we analyze the computational complexity.

6.1. Experiment settings

We use the two datasets, 24H and 1M, which were created in Wang et al. (2017). The performance parameters were collected through client-side service invocations. These selected services are commonly used services. The 24H dataset was collected by invoking the services for a day and the 1M dataset consists of performance parameters for a month. The prediction accuracy is measured using two metrics, which are defined as follows:

$$MAE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{\lambda}|\vec{p}_j - \vec{r}_j|}{N\cdot\lambda} \quad (29)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{\lambda}(\vec{p}_j - \vec{r}_j)^2}{N\cdot\lambda}} \quad (30)$$

where N is the number of experiments performed, and $\vec{p}$ is the online reliability time series of the component systems predicted by the experiment. $\vec{r}$ is the real reliability time series of the component system, λ is the length of the time series, which is the dimension of $\vec{p}$.

To demonstrate the effectiveness of the proposed approach, we compare it with a number of representative models: Average Value of Historical Reliability (Zheng and Lyu, 2010) (AVHR), Bayesian Rule based prediction (Csenki, 1990) (BR), and Regression Analysis based prediction (Andrzejak and Silva, 2007) (Reg). These models are described below:

1. AVHR uses collaborative filtering to make predictions and uses the average of component system reliability as the prediction result. In this paper, we take the average of historical reliability of the component system as the predicted value of each point in the online reliability time series.
2. BR predicts the probability distribution of the next error through the past error behaviors of the software system, and can effectively describe the evolution of the running state of the software system. In this paper, the historical reliability time series of component systems are clustered to obtain k reliability time series motifs $\{\widehat{P}_{r_i}|i = 1, 2, \ldots, k\}$. Based on this step, the paper researches and analyzes the transition relationship of these k reliability time series motifs. Then, we collect the reliability time series $\vec{O}_r$ for the recent time of the component system and mark the time series motifs (see Section 5.3), which is denoted as $\widehat{O}_r$. Finally, the $\widehat{P}_{r_i}$ with the maximum conditional probability $P(\widehat{P}_{r_i}|\widehat{O}_r)$ is considered as the online reliability time series of the component system.
3. Regression analysis (Reg) is a commonly used prediction method. In Andrzejak and Silva (2007), the least squares method is used to perform polynomial curve fitting. The prediction model is designed as:

$$\vec{y} = a_0 + a_1\vec{p}_r + a_2(\vec{p}_r)^2 \quad (31)$$

where $\vec{p}_r$ is the recent time period reliability time series of the component system. The loss function is $\sum_{i=1}^{\lambda}(\vec{r}_i - \vec{y}_i)^2$, $\vec{r}$ is the actual reliability time series, and λ is the length of the time series. The model parameters a_0, a_1 and a_2 are trained by historical reliability time series data of the component system. Finally, we take the reliability of the recent period component system into the model to compute $\vec{y}$, and consider $\vec{y}$ as the prediction result of the online reliability time series of the component system.

All experiments are conducted using a Windows 10 64-bit operating system, Intel(R) Core(TM) i7-6700 CPU 3.41GHz, 8GB of memory.

6.2. Results and analysis

In this section, we present the prediction performance, study the impact of different factors and then analyze the computational complexity of the proposed method.

6.2.1. Prediction performance

The prediction results of all the models are summarized in Tables 2–5. In these tables, N represents the number of predictions and k represents the number of time series motifs. The number of predictions for each method is 10, 20, 50, 100, 200, so we get 5

Table 2
MAE ($k = 20$).

Method	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$
m_MEMM	0.031	0.030	0.028	0.027	0.027
AVHR	0.095	0.091	0.085	0.093	0.087
BR	0.068	0.064	0.061	0.059	0.058
Reg	0.058	0.056	0.053	0.048	0.051

Table 3
RMSE ($k = 20$).

Method	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$
m_MEMM	0.064	0.057	0.056	0.054	0.054
AVHR	0.213	0.202	0.191	0.193	0.185
BR	0.137	0.121	0.112	0.108	0.107
Reg	0.112	0.098	0.097	0.093	0.094

Table 4
MAE ($k = 30$).

Method	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$
m_MEMM	0.030	0.030	0.027	0.026	0.026
AVHR	0.095	0.091	0.085	0.093	0.087
BR	0.065	0.063	0.061	0.058	0.058
Reg	0.058	0.056	0.053	0.048	0.051

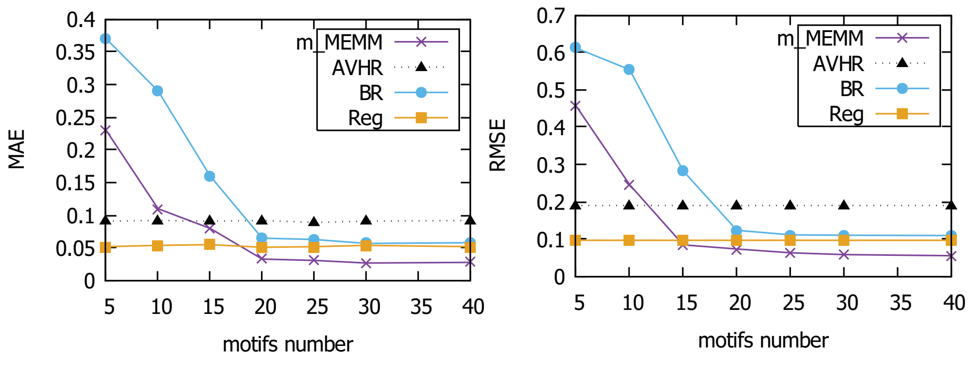


Fig. 9. $N = 50$, the effect of the number of motifs on the prediction result.

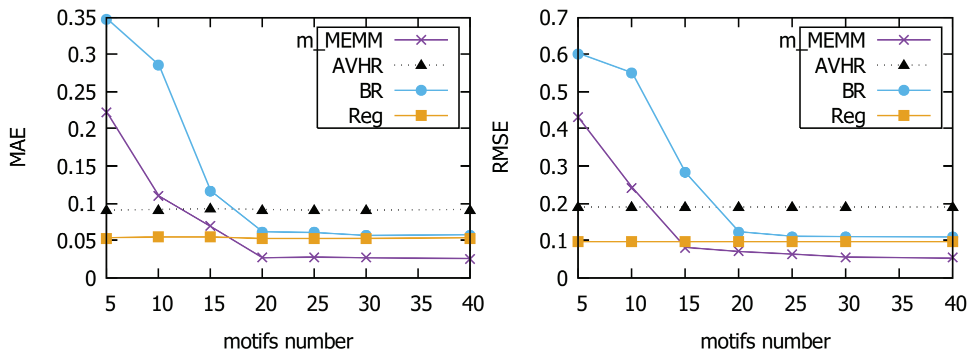


Fig. 10. $N = 100$, the effect of the number of motifs on the prediction result.

Table 5
RMSE ($k = 30$).

Method	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$
m_MEMM	0.062	0.057	0.055	0.053	0.053
AVHR	0.213	0.202	0.191	0.193	0.185
BR	0.136	0.121	0.112	0.107	0.106
Reg	0.112	0.098	0.097	0.093	0.094

kinds of experimental results. The numbers of time series motifs are 20 and 30, respectively.

It can be seen that as N increases, the prediction accuracy of the proposed method m_MEMM and BR is improved. The prediction accuracy of the AVHR and the Reg fluctuates. The reason that m_MEMM and BR achieve performance is that there are some regularities in the changes of the system parameter time series motifs in the component system, and these two methods can capture these regularities. In general, the proposed m_MEMM model achieves the best prediction performance, which makes it more suitable than other three methods for reliability prediction problems of component systems.

In addition, by comparing Tables 2 and 4 (or Tables 3 and 5), the number of time series motifs has no impact on AVHR and Reg prediction, indicating that the prediction results of AVHR and Reg have nothing to do with the number of time series motifs. For m_MEMM and BR, the accuracy of the predictions increases slightly with the increase in the number of time series motifs. In order to fully study the changes in the prediction results caused by changes in motifs number, we will discuss the effect of the number of time series motifs on the results of various prediction methods in next section.

6.2.2. The effect of number of motifs

In order to study the effect of the number of time series motifs on the prediction results, we vary the number of time series motifs from 5 to 40. For each reliability prediction method, we set the number of experiments N as 50 and 100 and observe the MAE and RMSE of all prediction methods. The experimental results are given in Figs. 9 and 10.

As shown in Figs. 9 and 10, the number of motifs does not affect the accuracy of AVHR and Reg. For m_MEMM and BR, the change in the number of motifs affects the accuracy of the prediction. Figs. 9 and 10 show that the accuracy of the two prediction methods gradually increase with the increase in the number of motifs, and the increasing rate slows down and becomes stabilized when the number of motifs reaches 20. In general, by setting a suitable number of time series motifs, m_MEMM achieves higher accuracy than the other three methods.

The number of motifs is a parameter to be set in time series motifs step in the reliability prediction step of the component system. It represents the k clusters of the clustering algorithm K-Means and reflects a certain feature of the time series. For the reliability time series motifs, they represent the running states of the component systems in the m_MEMM prediction model. In other words, the transition between the reliability time series motifs represents the transition between the running states of the component systems. The number of time series motifs indicates the number of running states of the component system. For a particular component system, it may have numerous running states for a long period of time. When the number of reliability time series motifs is too small, the model cannot fully describe all the running states of the component system and the transition relationship between different states. Thus, it is difficult to give a very good prediction accuracy. When the number of reliability time series motifs is too big, the model can describe all the running states of the component system and their transition relationship. However, the redundancy of the motifs will cause the model to become complex and the computational cost will increase. Therefore, a range of approximate motifs number can be selected first and then a fine tuning can be performed to reach the optimal value.

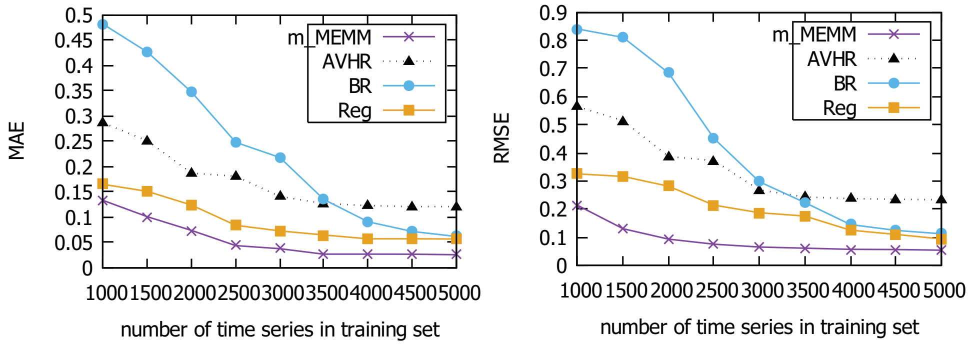


Fig. 11. The effect of the size of the dataset 24H on the prediction results.

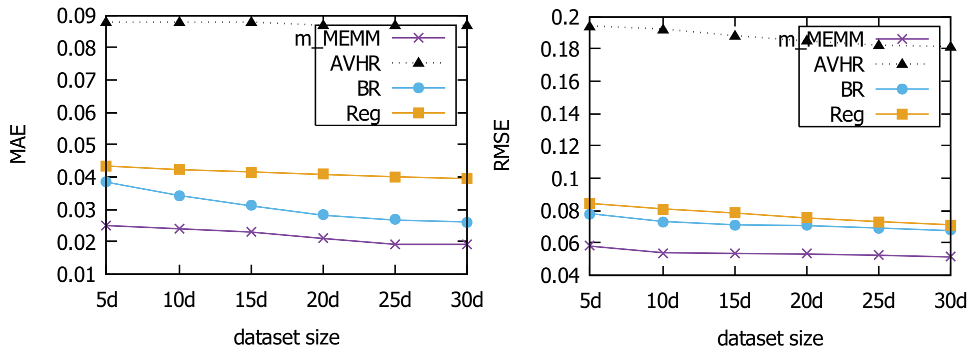


Fig. 12. The effect of the size of the dataset 1M on the prediction results.

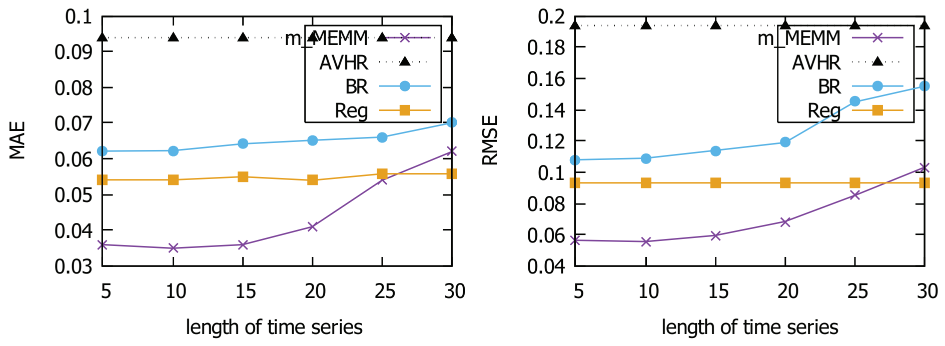


Fig. 13. The effect of time series length on prediction results on dataset 24H.

6.2.3. The effect of data set size

In this section, we will examine the effect of the dataset size on the accuracy of the prediction results. Here, we compare and analyze the experimental results of several prediction methods on the 24H dataset and the 1M dataset. As shown in Fig. 11, we extracted 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, and 5000 time series from the 24H dataset. The model is trained and reliability prediction is performed after the training.

From Fig. 11, we can see that the accuracy of AVHR, BR, and Reg is not high at the beginning due to the limited amount of training data. As the number of time series increases, the accuracy of these three methods is continuously improving. When the number of time series reaches 4000 or so, the accuracy of the three methods starts to stabilize. The proposed m_MEMM model achieves a higher prediction accuracy when the number of time series is small. As the number of time series increases, its prediction accuracy rate shows a small increase. As shown in Fig. 12, the experimental dataset 1M consists of one month of invocation data. We divide the dataset into 5-day, 10-day, 15-day, 20-day, 25-day, and 30-day scale. The results show that as the dataset size becomes larger, the prediction accuracy of several methods shows a small increase. In summary, we can conclude that m_MEMM is less sensitive to the dataset size than the other three prediction methods. In other words, it does not require a lot of sample data and has better performance. The reason is that the model combines the Maximum Entropy Model with Hidden Markov Model, which effectively leverages the advantages of both models. In particular, the Maximum Entropy Model can give a conditional probability distribution with maximum entropy for a small amount of sample data or when information is incomplete, and it sets the uncertain parts to equal probability (maximum entropy).

6.2.4. The effect of the length of time series

The choice of the length of the time series may also impact the final prediction result. To explore this effect, we set the length of the time series to 5, 10, 15, 20, 25 and 30 to examine the accuracy of different prediction methods. The results are shown in Figs. 13 and 14.

First of all, from the experimental results, with the changes in the length of time series, the prediction accuracy of AVHR and

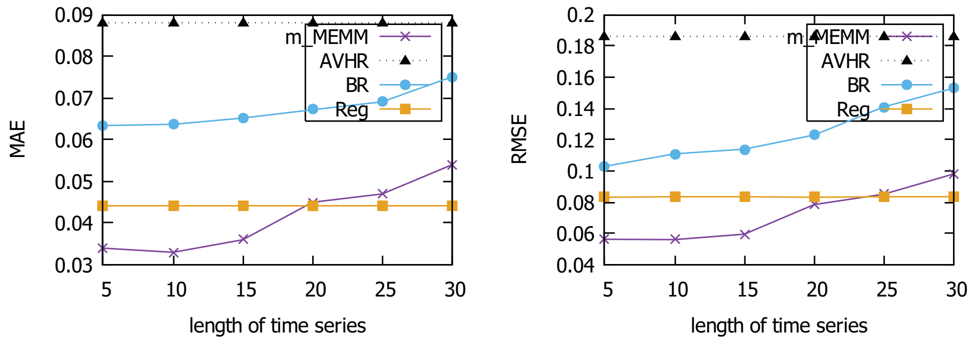


Fig. 14. The effect of time series length on prediction results on dataset 1M.

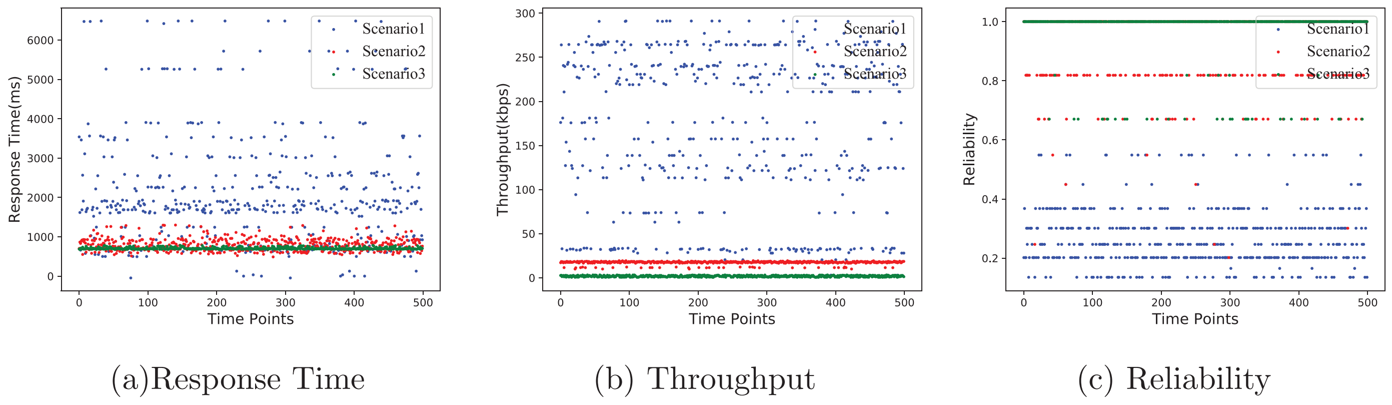


Fig. 15. The parameter comparison of the three scenarios.

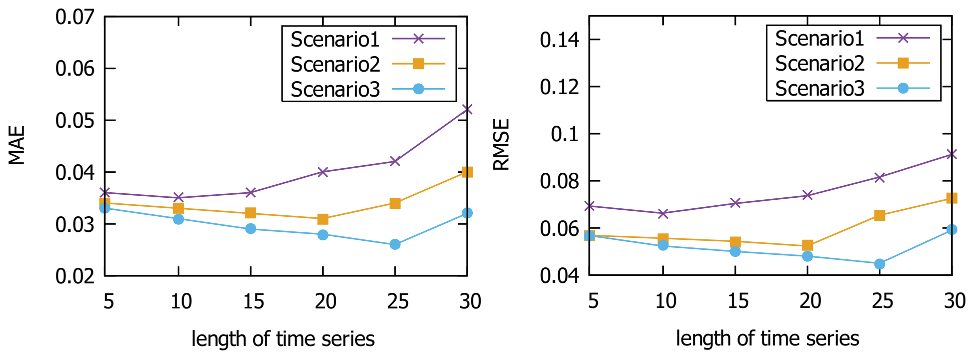


Fig. 16. The effect of time series length on prediction results in different scenarios.

Reg remains unchanged, indicating that the length of the time series has no effect on the accuracy of the AVHR and Reg prediction methods. Second, m_MEMM and BR show the same trend as the time series length changes. At the beginning, as the length of the time series increases, the accuracy of the two prediction methods gradually improves and then decreases. This is mainly due to the fact that the length of the time series is too short to fully describe a running state of the component system at a certain period of time. It may take several time series of this length to describe this kind of running state. Too long time series may describe two or more running states of a component system. These two situations eventually result in the model being unable to analyze the conversion relationship between different states of the component system. Finally, with a suitable time series length, the proposed m_MEMM model achieves the best prediction accuracy.

Furthermore, to help users select an appropriate length of time series, we chose three different services in the dataset to simulate the component systems invoked by different clients, where the time period selected for each service is 24 hours. These correspond to three different scenarios to show the effect of length of time series. Figs. 15 (a), (b), and (c) show the fluctuation of parameters over a period of time in 24 hours.

The experimental results are given in Fig. 16. As can be seen, with the increase of the length of time series, the prediction performance improves first and then becomes worse for all three scenarios. However, the optimal length of time series in different scenarios is different. A detail analysis shows that when the response time, throughput, and reliability parameters fluctuate with a higher frequency and larger amplitude, a shorter length of time series leads to a better prediction accuracy. Otherwise, a longer time series results in better prediction results. Therefore, if a user invokes a service with a high change frequency of parameters (e.g., in an environment with high network fluctuation), a shorter time series length is preferred. In contrast, if the user is in more stable network environment, the time series length can be increased accordingly to obtain better prediction results.

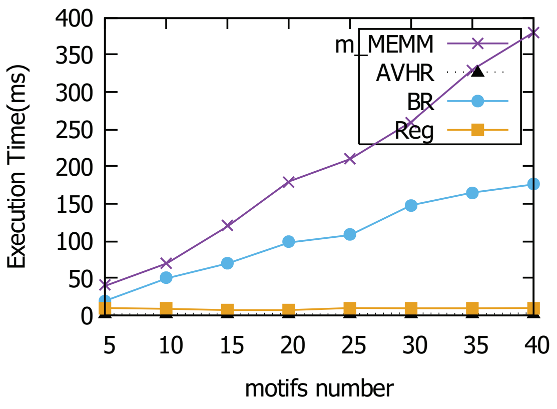


Fig. 17. Execution time comparison.

Therefore, in practice, the method proposed in this paper should consider the historical parameter variation regularities of the component system being invoked. The length of time series can be adjusted accordingly to obtain optimal prediction accuracy for online reliability time series prediction.

6.2.5. Computational complexity

The computational complexity of m_MEMM can be analyzed by considering two parts: model training (collecting system parameters and training the m_MEMM model) and reliability prediction (performing predictions according to the trained model and the current time system parameters). Since the training of the model is an off-line process, the training process has no impact on the computational complexity of the prediction process. Once the training process is completed, online reliability prediction can be performed, whose computational complexity is mainly related to the number of motifs.

To analyze the effect of the number of motifs on the computational complexity of reliability prediction, the 24H dataset was used with the number of motifs varying from 5 to 40 and the number of predictions as $N = 20$. The execution time was compared with other models.

As shown in Fig. 17, the execution time of m_MEMM and BR increases with the growth of motifs number. In contrast, the execution time of AVHR and Reg is almost constant. Among all the methods, m_MEMM is the slowest but its execution time is still acceptable for online reliability prediction. Furthermore, it achieves the highest accuracy, as indicated by Figs. 9 and 10. Therefore, when a user prefers a faster prediction speed, a small motif number should be selected considering its approximate linear relationship with the execution time. As a tradeoff, the prediction accuracy will decrease accordingly.

7. Conclusion and future work

In this paper, we study and analyze the historical system parameters of component systems of an SoS. The proposed approach exploits MEMM to model the running evolution of component systems. MEMM is a combination of a Maximum Entropy Model and an HMM, which effectively leverages the advantages of both. We further introduce the time series motifs, and propose a new prediction model (m_MEMM) that can be constructed in five steps: (1) data collection and preprocessing, (2) time series motifs calculation, (3) marking time series motifs, (4) parameter estimation, and (5) reliability prediction. We conduct a set of experiments to study the impact of different factors that affect the prediction accuracy of the model. Comparison with other prediction methods demonstrates the effectiveness of the proposed approach.

We identify three interesting future directions. Firstly, the MEMM model integrates two models, where the feature function is introduced into the Maximum Entropy Model. Features can be freely selected with the feature function, which can better reflect the relationship between the input variable x and the output variable y, and infer the component system's running state more accurately through observation data. We only define one type of feature function in this paper. More feature functions can be tested to further improve the prediction accuracy. Second, the prediction model requires a certain amount historical data from component systems to be trained before it can be used for prediction purpose. A more robust model should be able to quickly adapt to the changes of business requirement. Therefore, the predictive model needs to perform prediction while training the model. Finally, the first-oder Markov property is a strong assumption, which may affect the model accuracy. Higher order temporal dependencies may be explored in the future work to improve the model accuracy.

Acknowledgments

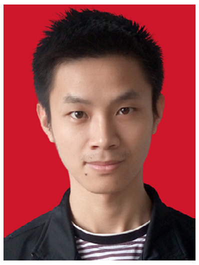
This work was partially supported by National Key Research and Development Plan (no. 2018YFB1003800) and NSFC Projects (nos. 61672152, 61532013), Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology. Qi Yu was supported in part by an NSF IIS award IIS-1814450 and an ONR award N00014-18-1-2875. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agency.

References

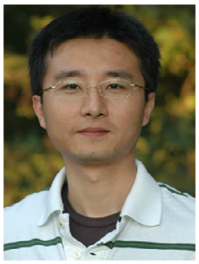
Amin, A., Colman, A., Grunske, L., 2012a. An approach to forecasting QoS attributes of web services based on Arima and Garch models. In: IEEE International Conference on Web Services, pp. 74–81.
Amin, A., Grunske, L., Colman, A., 2012b. An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 130–139.
Amin, A., Grunske, L., Colman, A., 2013. An approach to software reliability prediction based on time series modeling. J. Syst. Softw. 86 (7), 1923–1932.
Andrzejak, A., Silva, L., 2007. Deterministic models of software aging and optimal rejuvenation schedules. In: Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on. IEEE, pp. 159–168.
Baum, L.E., Petrie, T., 1966. Statistical inference for probabilistic functions of finite state Markov chains. Ann.Math.Stat. 37 (6), 1554–1563.
Baum, L.E., Petrie, T., Soules, G., Weiss, N., 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Stat. 41 (1), 164–171.
Berger, A.L., Pietra, V.J.D., Pietra, S.A.D., 1996. A maximum entropy approach to natural language processing. Comput.Linguist. 22 (1), 39–71.
Brosch, F., Koziolek, H., Buhnova, B., Reussner, R., 2012. Architecture-based reliability prediction with the palladio component model. IEEE Trans. Softw. Eng. 38 (6), 1319–1339. doi:10.1109/TSE.2011.94.
Buscarino, A., Corradino, C., Fortuna, L., 2018. Modeling and control of system of systems: the Tokamak scenario. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. doi:10.1109/ISCAS.2018.8351819.
Cheung, L., 2008. Early prediction of software component reliability. In: ACM/IEEE International Conference on Software Engineering, pp. 111–120.
Csenki, A., 1990. Bayes predictive analysis of a fundamental software reliability model. IEEE Trans. Reliab. 39 (2), 177–183.
Darroch, J.N., Ratcliff, D., 1972. Generalized iterative scaling for log-linear models. Ann.Math.Stati. 1470–1480.
Hall, J.W., Tran, M., Hickford, A.J., Nicholls, R.J., 2016. The Future of National Infrastructure: A System-of-systems Approach. Cambridge University Press.
Hoffmann, G.A., Trivedi, K.S., Malek, M., 2006. A best practice guide to resources forecasting for the apache webserver. In: Pacific Rim International Symposium on Dependable Computing, pp. 183–193.
Hoffmann, G.A., Trivedi, K.S., Malek, M., 2007. A best practice guide to resource forecasting for computing systems. IEEE Trans. Reliab. 56 (4), 615–628.
ISO/IEC/IEEE 24765:2010(E), 2010. Iso/iec/ieee international standard - systems and software engineering – vocabulary, pp. 1–418. doi:10.1109/IEEESTD.2010.5733835.
Kiciman, E., Fox, A., 2005. Detecting application-level failures in component-based internet services. IEEE Trans. Neural Netw. 16 (5). 1027–41.
Pfefferman, J.D., Cernuschi-Frias, B., 2002. A nonparametric nonstationary procedure for failure prediction. IEEE Trans. Reliab. 51 (4), 434–442.

Salfner, F., 2006. Modeling Event-driven Time Series with Generalized Hidden Semi–Markov Models.
Silic, M., Delac, G., Krka, I., Srbljic, S., 2014. Scalable and accurate prediction of availability of atomic web services. IEEE Trans. Serv. Comput. 7 (2), 252–264.
Silic, M., Delac, G., Srbljic, S., 2013. Prediction of atomic web services reliability based on k-means clustering. In: Joint Meeting on Foundations of Software Engineering, Esec/fse'13 / Meyer, Bertrand ; Baresi, Luciano ; Mezini, Mira, pp. 70–80.
Silic, M., Delac, G., Srbljic, S., 2015. Prediction of atomic web services reliability for qos-aware recommendation. Serv. Comput. IEEE Trans. 8 (3), 425–438.
Tekinerdogan, B., Erata, F., 2017. Modeling traceability in system of systems. In: the Symposium, pp. 1799–1802.
Vaidyanathan, K., Trivedi, K.S., 1999. A measurement-based model for estimation of resource exhaustion in operational software systems. In: International Symposium on Software Reliability Engineering, p. 84.
Wang, H., Wang, L., Yu, Q., Zheng, Z., Bouguettaya, A., Lyu, M.R., 2017. Online reliability prediction via motifs-based dynamic Bayesian networks for service-oriented systems. IEEE Trans. Softw. Eng. 43 (6), 556–579. doi:10.1109/TSE.2016.2615615.
Yu, Q., 2014. Qos-aware service selection via collaborative QoSevaluation. World Wide Web-internet Web Inf. Syst. 17 (1), 33–57.
Zheng, Z., Lyu, M.R., 2010. Collaborative reliability prediction of service-oriented systems. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. ACM, pp. 35–44.
Zhu, J., He, P., Zheng, Z., Lyu, M.R., 2014. Towards online, accurate, and scalable QoS prediction for runtime service adaptation. In: IEEE International Conference on Distributed Computing Systems, pp. 318–327.

Hongbing Wang is a professor from School of Computer Science and Engineering, Southeast University, China. He received his Ph.D. in computer science from Nanjing University, China. His research interests include Service Computing, Cloud Computing, and Software Engineering. He published more than fifty refereed papers in international conferences and Journals, e.g., Journal of Web Semantics, JSS, TSC, ICSOC, ICWS, SCC, CIKM, ICTAI, WI, etc. He is a member of the IEEE.



Huanhuan Fei is a postgraduate student in the School of Computer Science and Engineering, Southeast University, PR China. His research focuses on service selection and service composition.



Qi Yu received the Ph.D. degree in computer science from Virginia Polytechnic Institute and State University (Virginia Tech). He is an associate professor in the College of Computing and Information Sciences at the Rochester Institute of Technology. His current research interests lie in the areas of applied machine learning, data mining, and service computing. He has published over 80 papers, many of which appeared in top-tier venues in these fields. He is a member of the IEEE.

Wei Zhao is a postgraduate student in the School of Computer Science and Engineering, Southeast University, PR China. His research focuses on service selection and service composition.

Jia Yan is a postgraduate student in the School of Computer Science and Engineering, Southeast University, PR China. His research focuses on service selection and service composition.

Tianjing Hong is a postgraduate student in the School of Computer Science and Engineering, Southeast University, PR China. Her research focuses on service selection and composition and service prediction.