CARP: Context-Aware Reliability Prediction of Black-Box Web Services

Jieming Zhu[1], Pinjia He[1], Qi Xie[2]*, Zibin Zheng[3], Michael R. Lyu[1]

[1]Shenzhen Research Institute, The Chinese University of Hong Kong, China
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
{*jmzhu, pjhe, lyu*}*@cse.cuhk.edu.hk*

[2]School of Computer Science and Technology, Southwest Minzu University, China
qi.xie.swun@gmail.com

[3]Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education
School of Data and Computer Science, Sun Yat-Sen University, China
zhzibin@mail.sysu.edu.cn

***Abstract*—Reliability prediction is an important task in software reliability engineering, which has been widely studied in the last decades. However, modelling and predicting user-perceived reliability of black-box services remain an open research problem. Software services, such as Web services and Web APIs, generally provide black-box functionalities to users through the Internet, thus leading to a lack of their internal information for reliability analysis. Furthermore, the user-perceived service reliability depends not only on the service itself, but also heavily on the invocation context (e.g., service workloads, network conditions), whereby traditional reliability models become ineffective and inappropriate. To address these new challenges posed by blackbox services, in this paper, we propose CARP, a new contextaware reliability prediction approach, which leverages historical usage data from users to construct context-aware reliability models and further provides online reliability prediction results to users. Through context-aware reliability modelling, CARP is able to alleviate the data sparsity problem that heavily limits the prediction accuracy of other existing approaches. The preliminary evaluation results show that CARP can make a significant improvement in reliability prediction accuracy, e.g., about 41% in MAE and 38% in RMSE when only 5% of the data are available.**

***Index Terms*—Black-box services; reliability prediction; context awareness, matrix factorization**

I. Introduction

Reliability measures the probability of failure-free software operation for a specified period of time in a specified environment [12]. Reliability prediction is an important task in software reliability engineering [12], [13], which aims to predict failure rates of components and overall system reliability. These predictions are commonly used to evaluate design decisions, trade-off design factors, identify potential failure areas, and track reliability improvement [1]. In the last few decades, reliability prediction has been widely studied, producing a variety of prediction models (e.g., Palladio component model [3] and Poisson process model [7]). However, most of these existing models target at reliability analysis of traditional white-box software systems, where the reliability of system components are known or can be estimated through behaviour models from internal information of the components. How to model and predict the user-perceived reliability of emerging Web services remains an open reserach problem.

Nowadays, various software services such as Web services and Web APIs are emerging over the Internet. These services have become an integral part for building modern Web applications, in which each service provides a black-box functionality via some standard interfaces. To evaluate the reliability of a (third-party) black-box service, traditional white-box reliability prediction approaches become inapplicable due to a lack of its internal behaviour information. In addition, different from stand-alone software systems, software services operate over the Internet and likely serve different users spanning worldwide [40]. Therefore, the user-perceived reliability may differ from user to user due to different user locations, and vary from time to time due to dynamic service workloads and network conditions. In such a setting, it is more suitable to evaluate service reliability from user side than from system side as evaluating traditional software systems. As a result, modelling and predicting user-perceived reliability of black-box services is an important task, which is exactly the goal of our work.

Specifically, as with [19], [31], we compute user-perceived service reliability as the ratio of the number of successful service invocations against the total number of service invocations performed by the user. The most straightforward way, therefore, is to assess the reliability of a target service through real invocations from users. However, each service usually has many users and each user may need to assess a lot of alternative services (with similar or identical functionalities). Such exhaustive invocations can impose additional cost for users (e.g., the service invocations may be charged) and also incur expensive overhead for service systems (e.g., by consuming additional system resources), thus making it infeasible in practice. It is more desirable to identify approaches that

*The work was done when the author was visiting CUHK.

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.10

can achieve accurate reliability predictions without requiring additional service invocations.

Towards this end, a few initial efforts have been made by several recent studies, by applying the K-means clustering technique [19], [20] or collaborative filtering techniuqes used in recommender systems [31], [32]. These studies collect partial invocation data (i.e., observed reliability on the invoked services) from users and then construct statistical models for prediction of unknown reliability records. Whereas these approaches obtain encouraging results, two significant challenges remain: *1) Context modelling.* From a user's perspective, reliability not only depends on a service itself, but also is highly influenced by the context of service invocations (e.g., service load and network conditions). For example, user-perceived reliability may differ from user to user due to different user locations, and may vary from time to time incurred by service load variations and dynamic network conditions. How to leverage such context information to aid in reliability prediction is still a challenging problem. *2) Data sparsity.* In practice, each user typically invokes only a few services at each time, leading to a limited number of invocation samples. When modelling the reliability given a specific context, the data matrix becomes extremely sparse (i.e., most of reliability records are unknown). With limited training data, it is difficult to make accurate reliability predictions.

In this paper, we propose CARP, a context-aware reliability prediction approach that aims to tackle the above challenges. CARP models reliability as a function jointly determined by the (*user, service, context*) tuple of a service invocation. Then the model is constructed based on a novel formulation of context-specific matrix factorization by lerveraging the implicit context information between users and services. To guarantee computational efficiency, CARP comprises an *offline* step to train the context-aware reliability model from historical invocation data, and another *online* step to support on-demand reliability predictions for ongoing service invocations. We evaluate CARP on a publicly available dataset with real-world reliabiltiy samples collected from Amazon EC2 services [19]. The evaluation results show that CARP can better capture the characteristics inherent in reliability of black-box Web services and therefore yields a significant improvement in prediction accuracy (e.g., up to 41% in MAE and 38% in RMSE) over the state-of-the-art reliability prediction models.

The main contributions of this paper can be summarized as follows:

- We study the problem of predicting user-perceived reliability of black-box Web services, which remains an open and challenging research problem.
- We present CARP, a context-aware reliability model with its construction for reliability prediction, by leveraging a novel formulation of context-specific matrix factorization.
- The evaluation results show that CARP makes a significant improvement in prediction accuracy over the state-of-the-art reliability prediction models.

The remainder of this paper is organized as follows. Section II introduces the background. Section III describes the details of CARP. We report the evaluation results in Section IV and make some discussions in Section V. We then review the related work in Section VI, and finally conclude the paper in Section VII.

II. Background

Collaborative filtering (CF) techniques [21] are widely used to rating prediction in recommender systems, such as movie recommendation in Netflix. The goal of CF is to leverage partially-observed rating data to predict the remaining unknown ratings, so that movies can be recommended to users according to the predicted ratings. Matrix factorization (e.g., PMF [18]) is a classic model to address the collaborative filtering problem, which constrains the rank of the data matrix, *i.e.*, $rank(R) = d$. The low-rank assumption is based on the fact that the entries of R are largely correlated, thereby resulting in a low effective rank in R. Concretely, factoring a matrix is to map both users and services into a joint latent factor space of a low dimensionality d such that values of the data matrix can be captured as inner products of latent factors in that space. Then the latent factors can be employed for further prediction on unknown data entries.

Formally, given n users and m services, we denote latent user factors as $U \in \mathbb{R}^{d\times n}$ and latent service factors as $S \in \mathbb{R}^{d\times m}$. Both of them are used to fit the data matrix R, *i.e.*, $R \approx U^T S$. To avoid overfitting, regularization terms that penalize the norms of the solutions (*i.e.*, U and S) are added. Thus we resolve to minimize the following loss function:

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i^T S_j)^2 + \frac{\lambda_U}{2}\left\|U\right\|_F^2 + \frac{\lambda_S}{2}\left\|S\right\|_F^2, \quad (1)$$

where the first term indicates the sum of squared error in approximation. Especially, I_{ij} acts as an indicator that equals to 1 if R_{ij} is observed, and 0 otherwise. The remaining terms, namely regularization terms, are added to avoid overfitting. $\|\cdot\|_F$ denotes the *Frobenius norm* [18], and λ_U, λ_S are two parameters to control the extent of regularization.

Gradient descent is a widely used method to find a local minimum of an objective function in an iterative way. As for the PMF model expressed in Equation 1, the gradient descent algorithm works by initializing U_i and S_j randomly and iterating over the following updating rules until convergence:

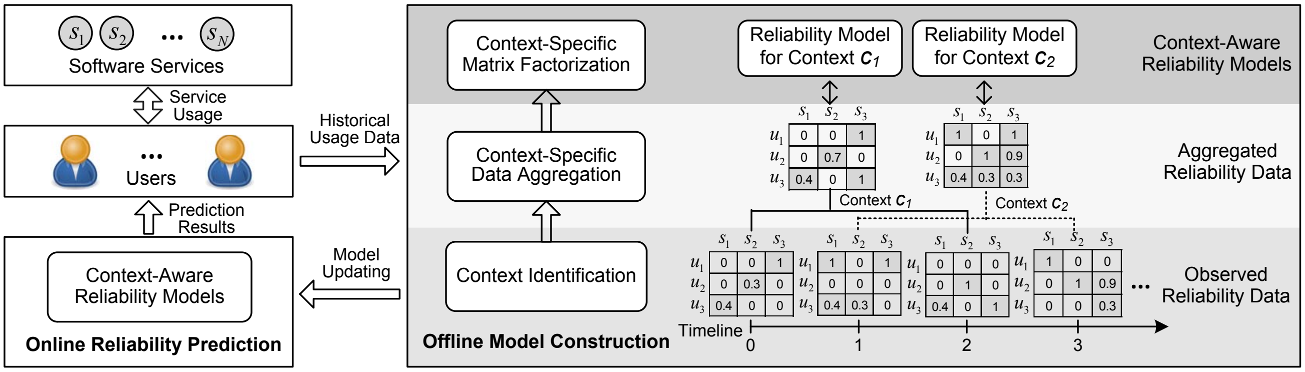
$$U_i \leftarrow U_i - \eta\frac{\partial\mathcal{L}}{\partial U_i}, \quad (2)$$

$$S_j \leftarrow S_j - \eta\frac{\partial\mathcal{L}}{\partial S_j}, \quad (3)$$

After obtaining the derivatives of U_i and S_j from Equation 1, we derive the following updating rules:

$$U_i \leftarrow U_i - \eta\big(\sum_{j=1}^{m} I_{ij}(U_i^T S_j - R_{ij})S_j + \lambda_U U_i\big), \quad (4)$$

$$S_j \leftarrow S_j - \eta\big(\sum_{i=1}^{n} I_{ij}(U_i^T S_j - R_{ij})U_i + \lambda_S S_j\big). \quad (5)$$


Software Services
s_1 s_2 ... s_N
Service Usage
Users
Prediction Results
Context-Aware Reliability Models
Online Reliability Prediction
Historical Usage Data
Model Updating
Context-Specific Matrix Factorization
Context-Specific Data Aggregation
Context Identification
Offline Model Construction
Reliability Model for Context c_1
Reliability Model for Context c_2
Context-Aware Reliability Models
Aggregated Reliability Data
Observed Reliability Data
Context c_1
Context c_2
Timeline
0
1
2
3
...

	s_1	s_2	s_3
u_1	0	0	1
u_2	0	0.7	0
u_3	0.4	0	1

	s_1	s_2	s_3
u_1	1	0	1
u_2	0	1	0.9
u_3	0.4	0.3	0.3

	s_1	s_2	s_3
u_1	0	0	1
u_2	0	0.3	0
u_3	0.4	0	0

	s_1	s_2	s_3
u_1	1	0	1
u_2	0	0	0
u_3	0.4	0.3	0

	s_1	s_2	s_3
u_1	0	0	0
u_2	0	1	0
u_3	0.4	0	1

	s_1	s_2	s_3
u_1	1	0	0
u_2	0	1	0.9
u_3	0	0	0.3

Fig. 1. The Framework of Context-Aware Reliability Prediction

In this way, the latent factors U_i and S_j move iteratively by a small step of the average gradients, *i.e.*, $\frac{\partial \mathcal{L}}{\partial U_i}$ and $\frac{\partial \mathcal{L}}{\partial S_j}$, where the step size is controlled by a learning rate η.

III. Context-Aware Reliability Prediction

In this section, we describe the overview and details of our context-aware reliability prediction approach.

A. Overview

Fig. 1 presents our context-aware reliability prediction framework, which comprises three phases: *1) Data collection*. A user-collaboration mechanism, proposed in our previous work [30], is applied to collecting historical usage data from users. Users can contribute their observed reliability data on the invoked services and get back personalized (i.e., from user side) reliability prediction results. *2) Offline model construction*. Using the collected reliability data, we can construct the context-aware reliability model by a process involving context identification, context-specific data aggregation, and context-specific matrix factorization. The model construction can be performed offline at a periodical interval to update the model parameters with newly-observed reliability data. *3) Online reliability prediction*. The constructed reliability models can be used to provide personalized reliability prediction results to users in an online manner.

B. Context-Aware Reliability Model

For traditional software systems, the researchers generally take reliability as a constant value that measures the probability of failure-free software operation. Given a specified period of time, the software reliability is defined as follows:

$$r(s), \tag{6}$$

where s denotes the specific software system or component. $r(s)$ depends on the software-specific parameters such as software architecture, system resources (e.g., CPU, memory, and I/O), and other software design and implementation factors.

However, this traditional reliability model is not applicable for measuring user-perceived reliability of black-box services. As mentioned before, service reliability should be evaluated from user side other than from system side as evaluating traditional software systems. Due to the influence of user locations and network connections, different users may experience quite different reliability even on the same service. To characterize user-perceived reliability, Zheng et al. [30], [31] propose the following model:

$$r(u, s), \tag{7}$$

where u and s denote the specific user and service respectively. $r(u, s)$ depends both on user u and service s.

Further, this model is extended to incorporate temporal information in [19], considering that user-perceived reliability may vary from time to time due to fluctuating service workloads and dynamic network conditions. Specifically,

$$r(u, s, t), \tag{8}$$

formulates the user-perceived reliability for an invocation $inv(u, s, t)$ between user u and service s at time slice t.

Although this new model, $r(u, s, t)$, can naturally characterize user-perceived reliability well, we find it difficult to directly apply it to reliability prediction. Because each user has limited historical usage data, applying $r(u, s, t)$ to model the data can lead to such data sparsity problem and thus result in inaccurate predictions.

In this paper, we argue that the time-dimensional characteristics can be typically captured by a finite set of context conditions, each of which is an abstract representation of the underlying factors such as service workloads and network conditions. It is further endorsed by the fact that service workloads and network conditions likely have regular daily distributions [23]. Thus, a specific context condition likely determines the reliability value at a specific time slice. Based on this observation, we propose a context-aware reliability model:

$$r(u, s, c), \tag{9}$$

where c denotes the specific context condition under which the invocations $inv(u, s, t)$ are performed. $r(u, s, c)$ indicates that the user-perceived reliability depends on the user u, the service s, and the context c. Especially, $r(u, s, t) \approx r(u, s, c)$, if the context condition is c at time slice t. With a limited number of contexts, this model can make reliability data less sparse. In the following, we will describe the use of this model for context-aware reliability prediction.

C. *Offline Model Construction*

Formally, we can collect a 3-dimensional matrix $R \in \mathbb{R}^{M\times N\times T}$, which records the reliability data for M users, N services, and T time slices. $R_{u,s,t} = r(u,s,t)$ when the reliability value $r(u,s,t)$ of invocations $inv(u,s,t)$ is observed; otherwise, we set $R_{u,s,t} = 0$ as an unknown entry. Due to the afore-mentioned data sparsity problem, the matrix R is highly sparse in practice with a large number of unknown entries. The goal of reliability prediction is to predict these unknown entries, whereby the reliability of an ongoing invocation can be further predicted. As illustrated in the right panel in Fig. 1, the offline model construction comprises three steps: *context identification*, *context-specific data aggregation*, and *context-specific matrix factorization*.

1) Context Identification

To characterize and identify different context conditions, we employ *k-means* clustering technique to cluster the reliability data R with T time slices into C clusters, where each cluster represents a specific context and different time slices grouped into one cluster belong to the same context. To achieve this, the observed reliability data between M users and N services at each time slice t can be constructed as a feature vector for *k-means* clustering. However, due to the sparse nature of R, the feature vectors would become high-dimensional and sparse, further leading to bad clustering performance. To overcome this issue, we define a feature vector f_t for time slice t using the average reliability value of each service:

$$f_t = \big(\bar{r}(s_1,t), \bar{r}(s_2,t), \cdots, \bar{r}(s_N,t)\big), \tag{10}$$

where $\bar{r}(s,t) = mean(\{R_{u,s,t} \mid R_{u,s,t} > 0,\ 1 \leq u \leq M\})$ calculates the average reliability value of service s over the observed entries at time slice t. Using these feature vectors, we perform data clustering and get C different context conditions.

2) Context-Specific Data Aggregation

Different time slices may be clustered into each context. To alleviate the data sparsity problem, we propose to aggregate the data of different time slices within the same context. An aggregated data matrix $\bar{R} \in \mathbb{R}^{M\times N\times C}$ can thus be obtained, where each entry $\bar{R}_{u,s,c}$ denotes the average reliability value between user u and service s in context c:

$$\bar{R}_{u,s,c} = mean(\{R_{u,s,t} \mid R_{u,s,t} > 0,\ t \in context\ c\}) \tag{11}$$

Especially, $\bar{R}_{u,s,c} = 0$ indicates that the reliability for invocations $inv(u,s,t)$ performed in context c is unknown. For example, in Fig. 1, the observed reliability data of four time slices are aggregated into two contexts (i.e., context c_1 and c_2) and thus the aggregated data become much denser.

3) Context-Specific Matrix Factorization

The problem of context-aware reliability prediction is to predict the unknown entries (where $\bar{R}_{u,s,c} = 0$) of the aggregated data $\bar{R}$. This can be modelled as a collaborative filtering (CF) problem, which aims for recovering the full matrix from a small number of observed entries. Taking Fig. 1 as an example, in the aggregated matrix for context c_1, we have four entries observed (e.g., $\bar{R}_{u_3,s_1,c_1} = 0.4$) and five unknown entries to predict (e.g., $\bar{R}_{u_1,s_1,c_1}$). Matrix factorization (MF) [18] is a classic CF model that allows for low-rank matrix approximation. Different with conventional matrix factorization, we have a 3-dimensional reliability data matrix $\bar{R} \in \mathbb{R}^{M\times N\times C}$, including one 2-dimensional M-by-N data matrix $\bar{R}^{(c)}$ in each context c ($1 \leq c \leq C$), where its entry $\bar{R}^{(c)}_{u,s} = \bar{R}_{u,s,c}$.

In such a setting, we propose context-specific matrix factorization. Formally, factorizing a data matrix $\bar{R}^{(c)} \in \mathbb{R}^{M\times N}$ is to map both users and services into a d-dimensional latent factor space, such that the values of $\bar{R}^{(c)}$ can be captured as the inner products of the corresponding latent factors $U^{(c)} \in \mathbb{R}^{d\times M}$ and $S^{(c)} \in \mathbb{R}^{d\times N}$, i.e., $\bar{R}^{(c)} \approx {U^{(c)}}^T S^{(c)}$, where ${U^{(c)}}^T$ is the transpose of $U^{(c)}$. Therefore, the context-specific MF model for context c is to minimize the following loss function:

$$\mathcal{L}^{(c)} = \frac{1}{2}\sum_{u,s} I^{(c)}_{us}\left(\bar{R}^{(c)}_{us} - {U^{(c)}_u}^T S^{(c)}_s\right)^2 + \frac{\lambda}{2}\left(\left\|U^{(c)}\right\|^2_F + \left\|S^{(c)}\right\|^2_F\right), \tag{12}$$

where the first term measures the sum of the squared errors between the observed value $\bar{R}^{(c)}_{us}$ and the estimated value ${U^{(c)}_u}^T S^{(c)}_s$, and the second is a regularization term used to avoid the overfitting problem [18]. $I^{(c)}_{us}$ acts as an indicator: $I^{(c)}_{us} = 1$ if $\bar{R}^{(c)}_{us} > 0$; $I^{(c)}_{us} = 0$, otherwise. λ is a parameter to control the extent of regularization.

The algorithm of *gradient descent* [18] is usually employed to solve the MF model in Equation 12. For ease of computation, we solve each context-specific MF model sequentially, and employ the solution of the last context for initialization of the current one (e.g., use $U^{(1)}$ and $S^{(1)}$ to initialize $U^{(2)}$ and $S^{(2)}$). At last, we can obtain a pair of $U^{(c)}$ and $S^{(c)}$ for each context c. In practice, the offline model construction can be performed periodically to update the models with newly-observed reliability data.

D. *Online Reliability Prediction*

The constructed models (i.e., $U^{(c)}$ and $S^{(c)}$) allow for reliability prediction for invocations performed between user u and service s in context c, i.e., $\hat{R}_{u,s,c} = {U^{(c)}_u}^T S^{(c)}_s$, where $\hat{R}$ denotes the predicted matrix corresponding to $\bar{R}$. This is the basis for performing online reliability prediction, which aims to predict the user-perceived reliability of an ongoing invocation $inv(u,s,t_c)$. Therefore, we seek to associate the invocation context at the current time slice t_c to an existing context c. In our implementation, we use the newly observed reliability data to help identify the current context. Specifically, given the observed feature vector $f_{t_c} = \big(\bar{r}(s_1,t_c), \bar{r}(s_2,t_c), \cdots, \bar{r}(s_N,t_c)\big)$, we group it into one of the existing context clusters. After obtaining the context c, the reliability of $inv(u,s,t_c)$, denoted as $\hat{r}(u,s,t_c)$, can be predicted by $\hat{r}(u,s,t_c) = \hat{R}_{u,s,c}$.

Statistics	Values
#Records	17,150
#Users	50
#Services	49
#Workloads	7
Reliability range	$0 \sim 1$
Reliability average	0.433

Fig. 2. Data Statistics

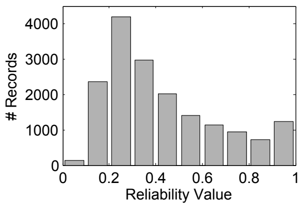


Fig. 3. Data Distribution

TABLE I
Accuracy Comparison

Metric	Approach	Data Density				
		5%	10%	15%	20%	25%
MAE	Baseline	0.176	0.170	0.168	0.169	0.170
	Hybrid [31]	0.152	0.084	0.079	0.073	0.073
	CLUS [19]	0.077	0.059	0.043	0.036	0.031
	PMF [32]	0.076	0.031	0.021	0.017	0.014
	CARP	**0.045**	**0.022**	**0.017**	**0.014**	**0.013**
	Impr.(%)	41.0%	27.8%	20.3%	13.9%	13.2%
RMSE	Baseline	0.217	0.211	0.210	0.210	0.211
	Hybrid [31]	0.204	0.109	0.102	0.094	0.094
	CLUS [19]	0.112	0.093	0.066	0.060	0.052
	PMF [32]	0.110	0.050	0.036	0.031	0.028
	CARP	**0.067**	**0.037**	**0.031**	**0.029**	**0.027**
	Impr.(%)	38.9%	24.9%	14.8%	5.8%	6.0%

IV. Evaluation

In this section, we present our results on evaluating the effectiveness of *CARP*. For ease of reproducing our approach, we release our source code with detailed experimental results on our WS-DREAM project page[1]. The WS-DREAM repository [36] is currently hosted on Github to disseminate our research results as well as to release open datasets and source code for Web service research. With both datasets and source code publicly released, our WS-DREAM repository would allow easily reproducing the existing approaches and give flexibility of extending new ones, which hopefully inspires more research efforts in the Service Society.

A. Data Description

Our experiments are conducted based on a real-world reliability dataset recently released in [19]. The dataset was collected using Amazon EC2 platform, which contains 17,150 reliability records from about 2.5 million invocations between 50 users and 49 services under 7 different workloads. Fig. 2 and 3 present some data statistics and the data distribution. Specifically, the services are implemented as matrix multiplication operations with different computational complexities, while the users are simulated by a "stress testing" tool, *loadUI* [19]. Both users and services are deployed into different locations across the seven EC2 regions. The service workload is controlled by setting different time intervals (i.e., 3∼9 sec) between consecutive invocations. Each reliability value is calculated as the successful ratio of 150 consecutive service invocations.

B. Evaluation Metrics

To evaluate prediction accuracy, we use two standard error metrics, MAE (Mean Absolute Error) and RMSE (Root Mean Square Error):

$$MAE = \sum_{inv(u,s,t)} \left|\hat{r}(u,s,t) - r(u,s,t)\right| \Big/ N, \qquad (13)$$

$$RMSE = \sqrt{\sum_{inv(u,s,t)} \left(\hat{r}(u,s,t) - r(u,s,t)\right)^2 \Big/ N}, \qquad (14)$$

where $r(u,s,t)$ and $\hat{r}(u,s,t)$ denote the observed reliability value and the corresponding predicted value respectively, for an invocation $inv(u,s,t)$. N is the total number of testing samples to be predicted. Both metrics measure the average magnitude of the errors and smaller values indicate better prediction accuracy. Compared to MAE, RMSE gives relatively high weights to large errors and turns to be more suitable when large errors are particularly undesirable. These two metrics have also been adopted by the existing work [19], [31].

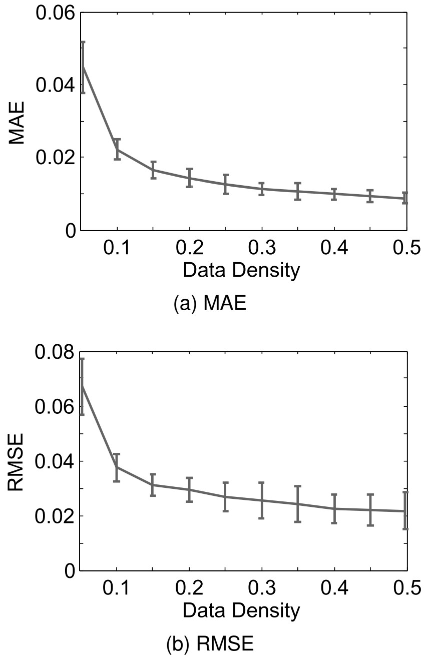
C. Accuracy Results

We compare *CARP* with the following state-of-the-art approaches that have been recently proposed for reliability prediction of Web services.

- **Baseline**: This is a baseline approach that simply uses the overall average value of the observed reliability data as prediction results.
- **Hybrid** [31]: This approach models reliability prediction as a collaborative filtering (CF) problem, which is solved by combining two traditional CF approaches: user-based approach (UPCC) and item-based approach (IPCC).
- **CLUS** [19]: Based on K-means clustering, this approach clusters historical reliability data according to user-specific, service-specific, and environment-specific parameters respectively, and hashes the average reliability value of each cluster for prediction.
- **PMF** [32]: This is a widely-used conventional matrix factorization model, where the reliability data are modelled by a pre-defined low-rank matrix model.

As we mentioned before, the observed reliability data are sparse in practice, because each user usually invokes only a small set of services out all of them. To simulate the data sparsity in our experiments, we randomly remove the entries from the data matrix R in our dataset, so that each user only keeps a few available historical reliability records. We use the remaining data for model construction and the removed values for accuracy evaluation. Specifically, we vary the data density from 5% to 25% at a step increase of 5%. Data density = 5%, for example, indicates that each user invokes only 5% of the services, and each service is invoked by 5% of the users. For *Hybrid* and *CLUS*, we employ the executable program with its parameters provided in [19]. For *PMF*, we carefully tune the parameters and set $d = 2$ and $\lambda = 0.01$ with best accuracy results. To make *CARP* consistent with other approaches, we set the number of context conditions $C = 7$ as with *CLUS*,

[1] http://wsdream.github.io/CARP



(a) MAE

(b) RMSE

Fig. 4. Impact of Data Sparsity

and set $d = 2$, $\lambda = 0.01$ as with *PMF*. Each experiment is run for 20 times and the average results are reported.

Table I provides the prediction accuracy results of different approaches in terms of MAE and RMSE. We can see that *CARP* consistently outperforms the other approaches with smaller MAE and RMSE. Compared to the most competitive results of *PMF*, *CARP* still has 13.2%∼41.0% improvement on MAE and 6.0%∼38.9% improvement on RMSE. It indicates that our *CARP* model fits the reliability data better with context-aware reliability modelling. In particular, CARP achieves a larger improvement at a smaller data density (e.g., the largest improvement is achieved at data density = 5%), which demonstrates its effectiveness in alleviating the data sparsity problem for reliability prediction.

D. *Impact of Data Sparsity*

To study the impact of data sparsity on prediction accuracy, we evaluate *CARP* by varying the data density from 5% to 50% at a step increase of 5%. A lower data density indicates higher data sparsity because more data are removed during data processing. Fig. 4 presents the evaluation results (with 95% confidence interval) on both MAE and RMSE. We can observe that better prediction accuracy can be achieved with the increase of data density from 5% to 50%. That is, MAE decreases from 0.045 to 0.009 and RMSE decreases from 0.067 to 0.022. The results show that more training data can usually provide more useful information for model construction and thus achieve better prediction accuracy. In particular, the significant fluctuation of the curve, when the data is extremely sparse (e.g., data density = 5%), further confirms that data sparsity is a great challenge in achieving accurate reliability prediction. Our *CARP* approach takes a first step forward for addressing the data sparsity challenge and achieves a significant improvement.

V. Discussion

Reliability measurement v.s. prediction. One may argue that active measurement (e.g., through invoking a service periodically) is a straightforward way to determine reliability of a service. However, this simple approach is not scalable. Each user may have a large number of services to measure, while each service usually has a large user base. Active measurement would incur prohibitive overhead to both users and services. Furthermore, many services are not free, which will lead to additional cost of service invocations. In these cases, it is desired to perform accurate reliability prediction without intensive direct service invocations.

Collaborative data collection. A data collection framework capable of assembling invocation records from users is needed to support online reliability prediction. Our work is developed based on underlying usage data collection, but we focus primarily on processing and prediction of reliability values. We have previously developed a WSRec framework [33] for collaborative QoS collection, where a set of QoS collector agents were developed using Apache Axis and further deployed on the global PlanetLab platform to collect QoS records of publicly-available Web services at runtime. This framework can be easily extended to support reliability data collection. In addition, a privacy-preserving scheme is explored in [38] by applying differential privacy techniques to dealing with potential privacy issues of usage data collection from different users. We also expect to employ a streaming data platform, e.g., Amazon Kinesis[2], to collect real-time usage data streams, but we leave the implementation of such an end-to-end system for future work.

Representativeness of datasets. CARP is a data-driven approach that highly depends on the characteristics of reliability data. We develop and validate CARP based on the reliability dataset collected from real-world services. But the results may still be limited by the diversity of our dataset. Real-world service usage datasets are scarce in public. To improve generalizability, we plan to further validate our apporach on some other QoS attributes, such as response time and throughput. We believe this work can serve as a good baseline for future research.

VI. Related Work

A. *Software Quality Assessment*

Software quality assessment [9], [22] is an important field of study and practice in software engineering that can aid in decision making during all phases of software lifecycle. Software quality [4], [8] generally covers many aspects of software products such as correctness, performance, reliability, maintainability, etc. Unfortuntely, many of them cannot be easily quantified or measured for software quality assessment.

[2] https://aws.amazon.com/kinesis

As a result, an abundance of software quality prediction models [9] have been built based on measurable internal metrics. Early work applies simple regression models (e.g., multivariate regression model [2], [14]) to establishing projections between various design structures and the resulting software quality charactersitics. More recent work (see [17]) proposes the use of sophisticated machine leraning techqniues for improving software quality prediction. These stuides focus on analyzing the expected software quality of design alternatives from the software itself.

In particualr, as an important aspect of software quality, software reliability assessment and prediction have been extensively studied in the last decades [12], [15]. The researchers have proposed a variety of reliability prediction models, such as Palladio component model [3], Poisson process model [7], structure-based model [5], etc. However, most of these existing models target at analyzing traditional white-box software systems, where the reliability of system components are all known. In this paper, we propose to address the problem of user-perceived reliability of black-box services, where existing models are inapplicable. We present a novel approach that can exploit historical usage data from users for context identification of service invocations and can further leverage them for context-aware reliability prediction.

B. QoS Prediction of Web Services

The work most closely related to ours is the study on QoS (Quality-of-Service) prediction of Web services. Web services are black-box software services that provide software components as building blocks for enterprise application integration (via Web service composition [26]). QoS attributes such as response time, throughput, and reliability are widely used to evaluate the non-functional aspects of Web services for QoS-based Web service composition [26]. To address the QoS prediction problem of Web services, some prior studies have proposed the use of collaborative filtering techniques in recent literature. Collaborative filtering (CF) [21] is a well-studied technique for rating prediction in recommender systems, which consists of two types of approaches: neighbourhood-based approaches and model-based approaches. For example, in our previous work we propose neighbourhood-based collaborative filtering approaches (e.g., UIPCC [31], [33]) and model-based collaborative filtering approaches (e.g., PMF [32], EMF [25], AMF [37], [39]) for service quality (or reliability) prediction. However, these models only consider user-specific and service-specific fators, which results in low prediction accuracy. Two more recent stuides [28], [29] further incorporate temporal information into their models and make use of tensor factorization for time-aware QoS prediction. But tensor factorization suffers from the scalability problem and is not sufficiently efficient for online reliability prediction in our setting.

Current research has seldom focused on user-perceived reliability prediction of software services. Zheng et al. [31] make the first effort in this direction, where they employ historical usage data from users for reliability prediction and model it as a collaborative filtering problem. In [31], they propose a neighbourhood-based approach, *Hybrid*, which combines two traditional CF approaches: user-based CF (UPCC) and item-based CF (IPCC). The following work [32] further extends a model-based approach, matrix factorization (PMF), to address this problem. However, these models only consider user-specific and service-specific parameters. Silic et al. [19] make a further step forward and incorporate environment-specific parameters for reliability prediction. This approach achieves scalability by clustering reliability data according to user-specific, service-specific, and environment-specific parameters, but sacrifices prediction accuracy (which is worse than *PMF* as shown in Table I). Our approach, instead, addresses these limitations on accuracy and scalability by performing context-aware reliability prediction.

C. Data-Driven Software Engineering

The data generated throughout the software lifecycle (e.g., source code, revision histories, bug reports, and runtime logs) contain a wealth of valuable information that can aid in software engineering tasks [34]. The goal to explore the potential of such rich data motivates a large body of studies related to mining software engineering data [24], software intelligence [6], and software analytics [27]. For example, Lessmann et al. study the use of classification models for defect prediction [10]. Lin et al. employ clustering techniques to help with system failure diagnosis [11]. Xie et al. employ natural language processing techniuqes to extract method specifications [16]. Zhou et al. leverage information retrieval techniques for bug localization [35]. All these studies employ data-driven approaches to gain actionable information and uncover powerful insights for better software development and maintenance. Our work can be viewed as another application in data-driven software engineering, where we describe the novel use of context-specific matrix factorization on historical invocation data for service reliability prediction.

VII. Conclusion and Future Work

This paper presents CARP, a context-aware reliability prediction approach for user-perceived reliability prediction of black-box services. CARP exploits historical usage data from users to assess the observed reliability of services, and further leverage them to construct context-aware reliability models. Through context-aware model training and prediction, CARP is capable of alleviating the data sparsity problem that heavily limits the existing models. The experimental results show that CARP makes a significant improvement in prediction accuracy over the state-of-the-art reliability models.

The use of data-driven approaches is promising for the quality management of black-box services in the field. We believe CARP can serve as a good starting point towards this end. As part of our future work, we plan to: *1)* develop more robust reliability prediction approaches to handle the data collection from malicious users and services, *2)* consider data privacy when performing collaborative reliability prediction, and *3)* perform reliability evaluations on real-world services to help identify and address reliability issues.

Acknowledgment

The work described in this paper was supported by the National Natural Science Foundation of China (Project No. 61332010 and 61502401), the National Basic Research Program of China (973 Project No. 2014CB347701), and the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14234416 of the General Research Fund).

References

[1] Reliability prediction basics. http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf.

[2] W. Agresti, W. Evanco, and M. Smith. Early experiences building a software quality prediction model. In *Proc. of the 15th Annual Software Engineering Workshop*, 1990.

[3] F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner. Architecture-based reliability prediction with the palladio component model. *IEEE Trans. Software Eng.*, 38(6):1319–1339, 2012.

[4] A. Gillies. *Software quality: theory and management*. Lulu.com, 3rd edition, 2011.

[5] S. S. Gokhale and M. R. Lyu. A simulation approach to structure-based software reliability analysis. *IEEE Trans. Software Eng.*, 31(8):643–656, 2005.

[6] A. E. Hassan and T. Xie. Software intelligence: the future of mining software engineering data. In *Proc. of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, pages 161–166, 2010.

[7] C. Huang, M. R. Lyu, and S. Kuo. A unified scheme of some nonhomogenous poisson process models for software reliability estimation. *IEEE Trans. Software Eng.*, 29(3):261–269, 2003.

[8] H.-W. Jung, S.-G. Kim, and C.-S. Chung. Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software*, 21(5):88–92, 2004.

[9] S. H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., 2nd edition, 2002.

[10] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Trans. Software Eng.*, 34(4):485–496, 2008.

[11] Q. Lin, H. Zhang, J. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In *Proc. of the 38th International Conference on Software Engineering (ICSE)*, pages 102–111, 2016.

[12] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. McGraw-Hill, Inc., 1996.

[13] M. R. Lyu. Software reliability engineering: A roadmap. In *Proc. of ACM/IEEE ICSE, Future of Software Engineering (FOSE) Track*, pages 153–170, 2007.

[14] J. Munson and Y. M. Khoshgoftaar. Regression modelling of software quality: Empirical investigation. *J. Electron. Mater.*, 19(6):106–114, 1990.

[15] G. J. Pai. A survey of software reliability models. *CoRR*, abs/1304.4539, 2013.

[16] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. M. Paradkar. Inferring method specifications from natural language API descriptions. In *Proc. of the 34th International Conference on Software Engineering (ICSE)*, pages 815–825, 2012.

[17] E. Rashid, S. Patnayak, and V. Bhattacherjee. A survey in the area of machine learning and its application for software quality prediction. *ACM SIGSOFT Software Engineering Notes*, 37(5):1–7, 2012.

[18] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proc. of Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[19] M. Silic, G. Delac, and S. Srbljic. Prediction of atomic web services reliability based on k-means clustering. In *Proc. of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 70–80, 2013.

[20] M. Silic, G. Delac, and S. Srbljic. Prediction of atomic web services reliability for QoS-aware recommendation. *IEEE Trans. Services Computing*, 8(3):425–438, 2015.

[21] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, 2009.

[22] T. Sunazuka, M. Azuma, and N. Yamagishi. Software quality assessment technology. In *Proc. of the 8th International Conference on Software Engineering (ICSE)*, pages 142–149, 1985.

[23] Y. Wang, W. M. Lively, and D. B. Simmons. Web software traffic characteristics and failure prediction model selection. *J. Comp. Methods in Sci. and Eng.*, 9(1,2S1):23–33, 2009.

[24] T. Xie, J. Pei, and A. E. Hassan. Mining software engineering data. In *Proc. of the 29th International Conference on Software Engineering (ICSE)*, pages 172–173, 2007.

[25] Y. Xu, J. Yin, Z. Wu, D. He, and Y. Tan. Reliability prediction for service oriented system via matrix factorization in a collaborative way. In *Proc. of the 7th IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 125–130, 2014.

[26] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327, 2004.

[27] D. Zhang and T. Xie. Software analytics: achievements and challenges. In *Proc. of the 35th International Conference on Software Engineering (ICSE)*, page 1487, 2013.

[28] W. Zhang, H. Sun, X. Liu, and X. Guo. Temporal QoS-aware web service recommendation via non-negative tensor factorization. In *Proc. of the International World Wide Web Conference (WWW)*, pages 585–596, 2014.

[29] Y. Zhang, Z. Zheng, and M. R. Lyu. Wspred: A time-aware personalized QoS prediction framework for web services. In *Proc. of the 22nd IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 210–219, 2011.

[30] Z. Zheng and M. R. Lyu. WS-DREAM: A distributed reliability assessment mechanism for web services. In *Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 392–397, 2008.

[31] Z. Zheng and M. R. Lyu. Collaborative reliability prediction of service-oriented systems. In *Proc. of the International Conference on Software Engineering (ICSE)*, pages 35–44, 2010.

[32] Z. Zheng and M. R. Lyu. Personalized reliability prediction of web services. *ACM Trans. Softw. Eng. Methodol.*, 22(2):12, 2013.

[33] Z. Zheng, H. Ma, M. R. Lyu, and I. King. QoS-aware web service recommendation by collaborative filtering. *IEEE T. Services Computing*, 4(2):140–152, 2011.

[34] Z. Zheng, J. Zhu, and M. R. Lyu. Service-generated big data and big data-as-a-service: An overview. In *Proc. of the IEEE International Congress on Big Data (BigData Congress)*, pages 403–410, 2013.

[35] J. Zhou, H. Zhang, and D. Lo. Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports. In *Proc. of the 34th International Conference on Software Engineering (ICSE)*, pages 14–24, 2012.

[36] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. WS-DREAM: Towards open datasets and source code for web service recommendation. Available at http://wsdream.github.io.

[37] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. Towards online, accurate, and scalable QoS prediction for runtime service adaptation. In *Proc. of the 34th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 318–327, 2014.

[38] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. A privacy-preserving QoS prediction framework for web service recommendation. In *Proc. of the IEEE International Conference on Web Services (ICWS)*, pages 241–248, 2015.

[39] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. Online QoS prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Trans. Parallel Distrib. Syst.*, preprint, 2017.

[40] J. Zhu, Z. Zheng, Y. Zhou, and M. R. Lyu. Scaling service-oriented applications into geo-distributed clouds. In *Proc. of the 7th IEEE International Symposium on Service-Oriented System Engineering*, pages 335–340, 2013.