Reliability Prediction for Service-Oriented System via Matrix Factorization in a Collaborative Way

Yueshen Xu, Jianwei Yin, Zizheng Wu, Dongqing He, Yan Tang
College of Computer Science and Technology
Zhejiang University
Hang Zhou, China
{xystjzcs, zjuyjw, zizhengwu, zjuhedq, allen_tang}@zju.edu.cn

Abstract—The reliability prediction of service-oriented system is a key problem, and has become increasingly important along with the wide utilization of service-oriented architecture. In this paper, we aim to improve the prediction accuracy of reliability in a collaborative way. First, we estimate the failure probability of each component through two independent models extended from Matrix Factorization. For each service and user, we identify the similar neighbors through similarity computation. Then, we build the service neighborhood-based MF model (SN-MF) and user neighborhood-based MF model (UN-MF). In the two models, each unknown failure probability is learned out assisted by similar neighbors’ historical failure records collaboratively. Further, we combine the two models together to build an ensemble model, and explicate the way of calculating the final failure probability of the whole system. Afterwards, the failure probability is mapped to reliability with a classical function. Finally, experiments conducted in a real-world dataset demonstrate the effectiveness of our models.

Keywords—Service-Oriented System, Software Failure, Reliability Prediction, Similarity Computation, Matrix Factorization

I. INTRODUCTION

Service-oriented architecture (SOA) is a software design pattern which has been widely used for constructing complex distributed computing systems on a general or specific purpose. In many typical cases, SOA combines a large chunk of Web services to connect and cooperate with each other in a network. Furthermore, it is independent of any business and technique details [14]. However, Web services are often provided by different organizations and become unreliable in uncertain Internet environment. So it is a significant challenge to predict the reliability of service-oriented systems, especially before the phase of real implementation and deployment.

Software reliability usually refers to the probability of uninterrupted service providing of a software system in a certain period under a given circumstance [11]. It has become one of the fundamental quality attributes of softwares. Many prediction approaches have been proposed for software reliability, which mainly utilize the historical operational records. In this paper, we focus on those approaches that conduct reliability prediction based on model building instead of software details. Traditionally, there are two kinds of reliability prediction methods for component-based system and service-oriented system. One focuses on system-level reliability prediction [2], [3], [4] and the other attaches importance to component-level reliability prediction [1], [5]. The two kinds of methods both assume that the reliabilities of components or the details of the given component have been already known. However, in most cases, such an assumption is not reasonable for service-oriented systems. On the one hand, it is hard to know the internal information of various services which are provided by different organizations. Usually, only those meta attributes that are provided through WSDL or other description files can be seen. On the other hand, even for the same service, different users can undergo different experiences of software reliability. That is because the stability of service-oriented system greatly relies on physical resources and Internet environment.

Unlike the previous approaches, a reliability prediction approach based on Collaborative Filtering (CF) algorithm was proposed in [20], [22], which utilized the historical failure records of a group of similar neighbors to predict the reliability of a certain service for an active user. Collaborative Filtering (CF) is one of the most widely used methods for personalized recommendation. It first identifies a part of similar users as the active user’s neighbors, which are based on the known ratings. Then, it will make recommendation on the basis of the predicted ratings [16]. However, it is worth noting that in recommender systems, whether a product is suitable for a user is decided by the user’s subjective preference. While for the service-oriented system, the reliability is not dominated by the user but by the objective condition, such as the stability of communication link and terminal device. To attack these challenges, we employ Matrix Factorization (MF) model in this paper.

To remedy the shortcomings of previous approaches, this paper proposes three collaborative prediction models to predict reliability for service-oriented systems. Our approaches aim at finding out those latent factors that can impact reliability. Some factors can measure those evident attributes, such as service type, provider and size. Other factors are some obscure attributes, for example, influence of communication link and bandwidth. Complementary to
most of previous approaches, which only focus on one aspect of component level or system level, our work takes both two levels into consideration. In detail, we propose service neighborhood-based MF model (SN-MF) and user neighborhood-based MF model (UN-MF). In the two models, the failure probability of each component is predicted from terminal-side and server-side respectively. Then, we combine the predicted results of the two models’ together to get comprehensive results. Afterwards, all component failure probabilities are utilized to calculate the final reliability of the service-oriented system which is usually implemented as an execution flow.

The contributions of this paper are three-fold as follows:

1) We propose three collaborative prediction approaches for service-oriented systems, all of which are more effective than traditional methods.
2) All of the three models can solve the problem of reliability prediction specific to both component level and system level.
3) We conduct sufficient experiments on a real-world dataset, which verify the effectiveness of our approaches.

The rest of this paper is organized as follows: Section II summarizes the related works. Section III explicates the two MF-based models, and then discusses different cases of system-level prediction. Section IV analyzes the experimental results, and Sect. V concludes the whole paper.

II. RELATED WORK

Over the past decades, many approaches to predicting software reliability have been proposed. Some early approaches are mainly specific to standalone software systems with using the past failure data [11]. More specifically, most of the approaches only focus on either component-level reliability or system-level reliability. Noteworthily, in recent years, several models based on collaborative filtering were proposed for predicting reliability of service-oriented systems from both of the two levels [20], [22].

A series of approaches [2], [3], [4], [6], [17] were proposed for system-level reliability analysis. Wang et al. [17] proposed an architecture-based reliability model, which predicted the system reliability involving in many factors, including each component’s reliability, system architecture and system operational records, etc. Goseva-Popstojanova et al. [6] predicted the reliability of a component-based software system based on the interaction between components and fault behavior of each component. Since most of these approaches are built under the assumption that the internal information and reliabilities of components are known, they are not suitable for services-oriented systems. In this paper, we will not depend on services’ internal information. Instead, we first conduct failure probability prediction for each single service by utilizing the historical failure records. Then, we will get the final reliability of the whole system according to the composite structures of service-oriented system.

Some other approaches [1], [5] focus on the reliability prediction in the component level. Cheung et al. [5] proposed a prediction framework for reliability specific to each component in the early phase of development period, employing as many resources as possible in that certain time. However, in the environment of service-oriented system, it is hard to know the internal information of the service and the service’s performance greatly relies on the unstable Internet environment. In contrast, our approaches try to find out the latent factors, for example, the stability of communication line, which can exert impact on the reliability in the real world.

III. RELIABILITY PREDICTION USING MATRIX FACTORIZATION

In this section, we first identify the similar neighbors through similarity computation. Then, we will explicate the service and user neighborhood-based MF models successively. Afterwards, we will discuss the way of combining the two models together. Finally, we will explain how reliability can be calculated according to failure probabilities.

A. Similarity Computation

For each user or service, the similar neighbors will be identified through historical failure records. We can calculate the failure probability of the invocation of each invoker to each service. Afterwards, all data of failure probability are arranged into the invoker-service matrix, which is a sparse matrix as shown in Fig. 1. In this matrix, each row represents an invoker, and each column represents a service. In this paper, the term invoker is used with the term user interchangeably. Those missing values need to be predicted.

![Invoker-Service Matrix](image)

For each two services $j$ and $h$, their similarity is calculated with adjusted cosine similarity [16]:

$$Sim(j, g) = \frac{\sum_{u \in U}(f_{uj} - \bar{f}_u)(f_{ug} - \bar{f}_u)}{\sqrt{\sum_{u \in U}(f_{uj} - \bar{f}_u)^2 \sum_{u \in U}(f_{ug} - \bar{f}_u)^2}},$$

(1)

where $u \in U$ represents all the users who invoked both service $j$ and $g$ before. $f_{uj}$ and $f_{ug}$ are failure probabilities of user $u$ invoking service $j$ and $g$, which are counted
with all the historical invocation records. \( f_{s_i} \) is the mean of all failure probabilities related with user \( u_i \). Similarly, the similarity between two users \( i \) and \( h \) is calculated as follows:

\[
\text{Sim}(i, h) = \frac{\sum_{s \in S} (f_{s_i} - \bar{f}_s)(f_{s_h} - \bar{f}_s)}{\sqrt{\sum_{s \in S} (f_{s_i} - \bar{f}_s)^2 \sqrt{\sum_{s \in S} (f_{s_h} - \bar{f}_s)^2}}. \tag{2}
\]

where \( s \in S \) contains all services that have been invoked by both user \( i \) and \( h \). \( f_{s_i}, f_{s_h} \) are the known failure probabilities, and \( \bar{f}_s \) is the mean failure probability of service \( s \).

Afterwards, for each service \( j \), those other services \( g \) whose similarities \( \text{Sim}(j, g) \) are in the \( K \) highest list will be selected out and compose service \( j \)'s neighborhood \( S_{TopK}(j) \) as \( \{ g \mid \text{Sim}(j, g) \in \text{the } K \text{ highest list of service } j \} \). Likewise, \( U_{TopK}(i) \) will also be constructed in the same way.

### B. Matrix Factorization

In this paper, Matrix Factorization is selected as the basic model for its success in recommender systems [9], [13], [21]. Let \( F = \{ f_{ij} \} \) (\( i = 1 \ldots M, j = 1 \ldots N \)) be the invoker-services matrix (see Fig. 1), where \( M, N \) represent the number of invokers and services. \( F \) can be factorized into two low-rank matrices approximately as follows:

\[
F \approx U^T S, \text{ or } f_{ij} \approx U_i^T S_j, \tag{3}
\]

where \( U, S \) are the user and service feature matrices respectively, and \( U \in R^{D \times M}, S \in R^{D \times N} \). \( U_i, S_j \) are the \( i \)th and \( j \)th column of \( U \) and \( S \), whose lengths are both \( D \), \( D \) is the number of latent factors. To minimize the difference between \( F \) and \( U^T S \), the square error between them is constructed. Besides, in practise, two regularization terms will also be appended to build the basic MF (Matrix Factorization) model as follows:

\[
\min_{U, S} \mathcal{L}(U, S) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}(f_{ij} - U_i^T S_j)^2 + \frac{\lambda_U}{2} \| U \|_F^2 + \frac{\lambda_S}{2} \| S \|_F^2, \tag{4}
\]

where \( I_{ij} \) is an indicator function whose value equals 0 if \( f_{ij} \) is unknown, and 1 otherwise. \( \| \cdot \|_F \) is Frobenius norm as the regularization term, which is used to forbid the overfitting problem during the learning process. \( \lambda_U \) and \( \lambda_S \) are small constants. The gradient descent algorithm can be used to achieve a local minimum of MF model.

### C. Service Neighborhood-based MF Model

For those services that have similar historical failure probabilities, they tend to have similar factors that can influence the reliability, such as network stability. Therefore, those services also tend to have similar latent features. To utilize such a similar relationship, the average predicted failure probability of user \( i \) to service \( j \)'s neighbors can be learned as

\[
\bar{f}_{U_{TopK}(i)} = \frac{1}{K} \sum_{g=1}^{K} U_i^T S_g. \tag{5}
\]

Then, inspired by [13], the predicted value of the active service is decomposed into two parts. One part is learned by its own feature vector, while the other part is the average predicted result gained through Eq. 5. The corresponding formula is shown as follows:

\[
f_{ij} \approx \alpha_U U_i^T S_j + \frac{(1 - \alpha_U)}{K} \sum_{g=1}^{K} U_i^T S_g, \tag{6}
\]

where \( \alpha_S \) is a regulatory factor to control the proportions, and \( g \in S_{TopK}(j) \) contains the selected similar neighbors of service \( j \). Then, the new objective function can be constructed as

\[
\min_{U, S} \mathcal{L}(U, S) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}(f_{ij} - \alpha_U U_i^T S_j + \frac{(1 - \alpha_S)}{K} \sum_{g=1}^{K} U_i^T S_g)^2 + \frac{\lambda_U}{2} \| U \|_F^2 + \frac{\lambda_S}{2} \| S \|_F^2. \tag{7}
\]

Now, we build the service neighborhood-based Matrix Factorization model (SN-MF). Then, the gradient descent algorithm can be also used in an iterative manner to obtain the local optima of \( U_i \) and \( S_j \).

### D. User Neighborhood-based MF Model

In the terminal user side, similarly with the server side, those users whose associated failure probabilities are similar also tend to encounter similar influencing factors that impact the reliability of SOA systems. To employ such kind of similar relation, at first, we calculate the average predicted result of \( U_{TopK}(i) \) as

\[
\bar{f}_{U_{TopK}(i)} = \frac{1}{K} \sum_{h=1}^{K} U_h^T S_j. \tag{8}
\]

Afterwards, like in [21], we also divide the predicted value into two parts, which are learned from the active user and his similar neighbors. However, in this paper, the similarity between two users are calculated with adjusted cosine similarity (see Eq. (2)), and the original model is simplified into the following form in which it is not necessary to calculate the weight.

\[
f_{ij} \approx \alpha_U U_i^T S_j + \frac{(1 - \alpha_U)}{K} \sum_{h=1}^{K} U_h^T S_j, \tag{9}
\]

where \( \alpha_U \) is a factor that regulates the weights of the two parts. \( h \in U_{TopK}(i) \) represents each similar neighbor of
user $i$. Then, we can build the objective function of user neighborhood-based Matrix Factorization model (UN-MF):

$$
\min_{U, S} L(U, S) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}(f_{ij} - (\alpha_U U^T_i S_j + (1 - \alpha_U) \frac{1}{K} \sum_{h=1}^{K} U^T_h S_j))^2 + \frac{\lambda_U}{2} \Vert U \Vert \|F \|_F^2 + \frac{\lambda_S}{2} \Vert S \Vert \|F \|_F^2.
$$

(10)

**E. Ensemble Model**

In this section, we combine the prediction results of UN-MF and SN-MF together to build a unified model. Concretely, we employ Multivariate Linear Regression to predict an ensemble result as follows:

$$
f_{ij} \approx \beta_1 \times f_{UN}(i, j) + \beta_2 \times f_{SN}(i, j) + \beta_0,
$$

(11)

where $f_{UN}(i, j)$, $f_{SN}(i, j)$ are the predicted failure probabilities of UN-MF, SN-MF respectively. $\beta_0, \beta_1, \beta_2$ are the coefficients in multivariate linear regression. Then, we can build the corresponding objective function according to the least square method as

$$
\min L = \sum_{(i, j)} (f_{ij} - \beta_1 f_{UN}(i, j) - \beta_2 f_{SN}(i, j) - \beta_0)^2,
$$

(12)

where $\beta_0, \beta_1, \beta_2$ can be learned out through matrix manipulation or gradient descent algorithm. We name this model as Ensemble Matrix Factorization (E-MF) model.

**F. System-level Reliability Prediction**

After the prediction for each component, the failure probability of the whole system can be calculated by dissecting its composite structure. There are several typical composite structures in service-oriented systems [8], [18], four of which are shown in Fig. 2. For more information, you can refer to Jaeger et al. [8] or Yu et al. [18]. In this paper, we hold the assumption that the failure probability of each component is independent [19]. Also, we do not involve in the case that some software faults will have been fixed along with time. Each type of standard structure owns its fixed way of combining the failure probability of each service together. For example, for the sequence structure (see Fig. 2(a)), the probability of successful running is $\prod_{i=1}^{L} (1 - f_i)$, where $f_i$ is the failure probability of each component, and $L$ is the number of standalone components. Hence, the whole failure probability $f$ is

$$
f = 1 - \prod_{i=1}^{L} (1 - f_i).
$$

(13)

Similarly, the failure probability of the Split XOR structure can be calculated as $f = 1 - \sum_{i=1}^{L} p_i (1 - f_i)$, where $p_i$ is the probability that one branch will be operated. For other standard formulas, you can refer to Zheng et al. [20]. Besides, for those service-oriented systems whose structures are not so typical, their system-level failure probabilities can be calculated in a hierarchical way [22]. The whole structure is first divided into different submodules whose structures are standard, and then the failure probabilities of those submodules can be calculated. Finally, the whole failure probability can be obtained through combination.

Figure 2. Composite Structures in Service-Oriented System

There are many mapping functions between reliability and failure probability [12]. We take one of the most widely used function for instance as follows:

$$
R(T) = exp(-f \times n \times T),
$$

(14)

where $n$ is the times of operation in per time unit, and $T$ is the time duration. In practice, the predicted results of SN-MF and UN-MF can be also used directly without ensemble.

**IV. EXPERIMENTS**

In this section, we conduct experiments to measure the prediction accuracy of our approaches, focusing on answering the following questions: 1) How do our approaches perform compared with other algorithms? 2) What are the optimal parameters, including $\alpha_S$, $\alpha_U$ and $K$, to control the performance of our approaches?

**A. Evaluation Metric**

In this paper, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are utilized to measure the difference between failure probabilities that are predicted and actually counted as follows:

$$
RMSE = \sqrt{\frac{1}{N} \sum_{i,j}^N (f_{ij} - \hat{f}_{ij})^2},
MAE = \frac{1}{N} \sum_{i,j}^N |f_{ij} - \hat{f}_{ij}|,
$$

where $f_{ij}$ and $\hat{f}_{ij}$ are the real and predicted values respectively, and $N$ is the total number of failure probabilities to be predicted.

**B. Dataset**

Our experiments are conducted on a real-world dataset containing 15000 response records collected by Zheng et al. [20]. In the preprocessing phase, the original data are transformed into the invoker-service matrix (see Fig. 1).
C. Prediction Accuracy Comparison

Several state-of-the-art methods are picked as benchmarks including:

1) UPCC: this is a classical user-based approach that utilizes the weighted average failure probability of similar users to predict missing values.

2) IPCC: this is a classical item-based approach that utilizes the weighted average failure probability of similar services.

3) Hybrid-PCC: this is the linear mixture of UPCC and IPCC with weights.

4) Basic MF: this is the basic Matrix Factorization model.

In the invoker-service matrix, the training dataset is generated by randomly hiding known values. Those hidden values compose the testing set, while the corresponding remaining data compose the training set. In this paper, the training set density ranges from 5% to 20% in the interval step of 2.5%. The default parameter setting is that $S = 0.8$, $U = 0.2$, $K = 10$ and $D = 10$.

As shown in Table I, both SN-MF and UN-MF outperform all PCC-based methods and Basic MF model, which verifies the effectiveness of collaborative utilization of similar neighbors. Furthermore, UN-MF is always superior to SN-MF, which indicates that the failure records of similar user neighbors are more valuable than those of service neighbors’. Meanwhile, E-MF performs consistently better than SN-MF and UN-MF in all cases, which indicates the effectiveness of results ensemble. Besides, for all algorithms, RMSE values steadily decline as the training set gets denser, since more training data can convey more invocation experiences to learn the failure probability more accurately.

### Table I

<table>
<thead>
<tr>
<th>Approach</th>
<th>TD=5%</th>
<th>TD=7.5%</th>
<th>TD=10%</th>
<th>TD=12.5%</th>
<th>TD=15%</th>
<th>TD=17.5%</th>
<th>TD=20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>RMSE</td>
<td>RMSE</td>
<td>RMSE</td>
<td>RMSE</td>
<td>RMSE</td>
<td>RMSE</td>
</tr>
<tr>
<td>UPCC</td>
<td>0.18862</td>
<td>0.18104</td>
<td>0.17542</td>
<td>0.17184</td>
<td>0.16665</td>
<td>0.16347</td>
<td>0.15929</td>
</tr>
<tr>
<td>IPCC</td>
<td>0.18708</td>
<td>0.17859</td>
<td>0.17235</td>
<td>0.16722</td>
<td>0.16056</td>
<td>0.15732</td>
<td>0.15204</td>
</tr>
<tr>
<td>Hybrid-PCC</td>
<td>0.18693</td>
<td>0.17836</td>
<td>0.17195</td>
<td>0.16679</td>
<td>0.15999</td>
<td>0.15655</td>
<td>0.15120</td>
</tr>
<tr>
<td>Basic MF</td>
<td>0.16368</td>
<td>0.16048</td>
<td>0.15783</td>
<td>0.15446</td>
<td>0.15232</td>
<td>0.14764</td>
<td>0.14596</td>
</tr>
<tr>
<td>SN-MF</td>
<td>0.15735</td>
<td>0.15309</td>
<td>0.14425</td>
<td>0.13444</td>
<td>0.13689</td>
<td>0.13460</td>
<td>0.13442</td>
</tr>
<tr>
<td>UN-MF</td>
<td>0.11730</td>
<td>0.11256</td>
<td>0.10643</td>
<td>0.10441</td>
<td>0.10310</td>
<td>0.10199</td>
<td>0.09885</td>
</tr>
<tr>
<td>E-MF</td>
<td>0.11626</td>
<td>0.10958</td>
<td>0.10607</td>
<td>0.10426</td>
<td>0.10299</td>
<td>0.10040</td>
<td>0.09532</td>
</tr>
</tbody>
</table>

E. Impact of K

The parameter $K$ determines the number of similar neighbors who are integrated in the learning process. We investigate the impact of $K$ in the range of 5 to 40, under the default parameter setting.

As it can be seen in Fig. 4, for both SN-MF and UN-MF, the prediction accuracy almost remains invariant in the whole range. It indicates that only a small number of service or software system, the failure probability is mainly determined by its running stability and robustness, so the learning result of the service itself accounts for a much large portion. By contrast, in the real scenario of software operation or service invocation, a user usually cannot control the reliability in terminal side. Therefore, they have to rely on their neighbors’ predicted results to a large extent.

![Impact of α₀, α₀ (K = 10, D = 10)](image)

![Impact of K (α₀ = 0.8, α₀ = 0.2, D = 10)](image)
neighbors are enough to obtain relatively high accuracy. Besides, UN-MF can achieve better performance than SN-MF consistently, which indicates that the similar user neighbors can provide more valuable reference information.

V. CONCLUSION AND FUTURE WORK

In this paper, to predict the reliability of service-oriented system, we first build two collaborative prediction models from the aspects of server and terminal respectively to predict the failure probability of each component in the whole system. In the two models, the unknown probability is learned with the assistance of similar neighbors, which are selected through similarity computation. Then, we combine the two models together into an ensemble model. Afterwards, by the way of dividing the whole system into several standard composite structures, we calculate the system-level failure probability by aggregation. Then, we introduce a typical function to transform failure probability into reliability. Finally, we conduct comprehensive experiments on the real-world dataset, which verify the effectiveness of our methods.

Though we have evaluated our models in the real-world dataset, we have not applied them in a real system. Hence, in the future, we plan to test our models in a service-oriented system. Wherein, we will seek solutions to solve the prospective problems, such as service composition.

ACKNOWLEDGMENT

This paper is fully supported by National Natural Science Foundation of China under Grant (No.61272129), National High-Tech Research Program of China (NO. 2013AA01A213), New-Century Excellent Talents Program by Ministry of Education of China (No.NCET-12-0491), Zhejiang Provincial Natural Science Foundation of China (LR13F020002), Science and Technology Program of Zhejiang Province (No.2012C01037-1).

REFERENCES


