Estimating the Reputation of Newcomer Web Services Using a Regression-Based Method

Okba Tibermacine[a], Chouki Tibermacine[b], Foudil Cherif[a]

[a]*Biskra University, LESIA, P.B. 145 R.P, Biskra 07000, Algeria*
[b]*LIRMM, CNRS and Montpellier University, France*

Abstract

In this paper, we propose a novel method to estimate the initial reputation values of newcomer web services. In fact, the reputation of web services is one of the criteria used for recommending services in service-oriented computing environments. The lack of evaluating the initial reputation values can subvert the performance of a service recommendation system making it vulnerable to different threats like whitewashing and Sybil attacks, which negatively affect its quality of recommendation. The proposed method uses Quality of Service (QoS) attributes from a side, and reputation values of similar services from the second side, to estimate the reputation values of newcomer services. Basically, it employs regression models, including Support Vector Regression, in the estimation process of the unknown reputation values of newcomers from their known QoS values. We demonstrate the efficiency of the method in estimating the reputation of newcomer services through statistical evidences gathered from experimentations conducted on a set of real-world web services.

Keywords: Web services recommendation, reputation measurement, support vector regression, feedback rating, honest and malicious service raters.

Email addresses: o.tibermacine@univ-biskra.dz (Okba Tibermacine), Chouki.Tibermacine@lirmm.fr (Chouki Tibermacine), foud_cherif@yahoo.fr (Foudil Cherif)

1. Introduction

Web Service recommendation systems (WSRSs) provide a precious assistance to users in selecting the best available Web Services (WS) to implement their business processes. To recommend services, WSRSs manage different kinds of metrics related to services, among which **reputation**. This subjective quality metric is an aggregation of feedback ratings gathered from service users. It reflects how users perceive this service, which is a good indicator of its Quality of Experience (QoE). The management of reputation plays a significant role in such systems. Recently, many reputation management models have been proposed to accurately evaluate the reputation of services [1–8]. Although these models have addressed many aspects in reputation evaluation such as user credibility, time sensitivity, personalized preferences, majority ratings, the evaluation of **newcomer** services with the absence of feedback ratings is still an important aspect that it has not been tackled thoroughly.

Indeed, assigning reputation values to newcomer (newly published and/or never used) services that have an empty rating history is an important and challenging issue due to the following reasons:

- WSRSs have to assign fair reputation values to newcomer services in order to enhance their visibility to users and to give these services a chance to compete with longstanding similar services during service selection phases. Thereby, the system provides a solution to the "cold start" problem, which describes the situation in which a recommender system is unable to make a meaningful recommendation due to an initial lack of users feedback-ratings [9].

- Initial reputation value, attributed by a WSRS, has to reflect the non-functional characteristics (QoS values) of a particular service. This value should not be general values related to the state of the recommender system itself, such as the average reputation value assessed in the system to all newcomer services [10], or assigning a fixed reputation value based on the rate of maliciousness in the system such as proposed in [11] as an elegant solution to the aforementioned problem.

- The lack of a correct estimation of initial reputation values of newcomer services may subvert the performance of the WSRS itself, making it vulnerable to different threats [12] (e.g., the Sybil attack [13]).

- WSRSs have to provide a solution to the *Whitewashing* problem too [14]. Whitewashing (i.e. changing the identity of a malicious user/service in the system) occurs when an entity leaves the recommendation system, then it reintegrates itself in the system with a new identity in order to erase its poor reputation acquired with its previous identity [11].

Though some notable solutions have been proposed in the literature for evaluating the reputation of newcomer services (e.g. [10, 11, 15–17]), most of these solutions assign the same initial reputation value to every newcomer service. Or, they do not offer a complete solution that addresses all the previous challenges.

In this paper, we propose a new model that refines and completes our initial proposition [18] to estimate initial reputation values of newcomer services in WSRSs. A correct estimation of these values allows better recommendations, thus a better help in selecting web services that satisfy clients' requirements.

Even though reputation is a subjective measure, it reflects users' satisfaction about a service's offered functionality and Quality of Service. It has been observed that fair feedback ratings provided by the majority of honest users are correlated, even with a slight deviation (due to differences between raters' opinions), with the QoS of used services [19]. Hence, we use QoS and reputation data of longstanding services to build a reputation estimation model for bootstrapping the reputation of newcomer services. We mean by "longstanding services" the services that have long feedback records constructed from collected user feedback ratings.

In fact, QoS and reputation values could have a linear or nonlinear correlations. Thus, we employ in our propositions both (i) a Linear Regression Model as a reputation estimator to deal with linear relationships, and (ii) a Support Vector Regression (SVR) model as a general reputation estimator to deal with general cases (that is, cases with linear or nonlinear relationships between reputation values and QoS).

Though the model that we propose considers Web services as the "first-class citizens" of the recommendation system, it can be safely used with any kind of SaaS (*Software as a Service*), whether it is a Cloud, REST/RESTful/WSDL or mobile service, among others. The word service can also be treated as an API in its general sense. We focused in this work on Web services, because we are convinced that a lot of software systems are deployed in the Web, and these still need the use of Web services or Web APIs for manag-

ing machine-to-machine interactions, whether these are WSDL/SOAP-based, which at the time of this writing are receiving less and less attention, REST-based, WebSocket-based, or any other protocol or architecture style usable over HTTP, which is currently the standard protocol for the Web. The most important aspect in our work is that a service has a set of functionalities (described by its iterface, which can be WSDL-based, JSON-based, or any other processable/parsable interface), and a set of Qualities (QoS) which are measurable. A service can compete with other services and it can be recommended based on its reputation which is evaluated by aggregating user feedbacks.

Basically, our reputation estimation method is built on three main phases:

1. **Provider reputation evaluation:** The system assesses the reputation of the new service's provider from reputation values of its previously published web services. Then, provider's reputation is employed in bootstrapping the reputation of the newcomer service.
2. **Similarity-based estimation of newcomer's reputation:** In this phase, the system selects among long-standing web services those which are functionally similar to the newcomer service. Then, the system selects top-K neighbor services that have high QoS value correlation. Finally, the system evaluates the reputation of the newcomer service based on its neighbors' reputation values (Technique 1), or using a multiple regression model (Technique 2) that is built from i) reputation values and ii) QoS values of the service's similar neighbors.
3. **Support Vector Regression-based estimation of newcomers reputation:** In this phase, the system deals with general cases (e.g. new services published by a new provider with no similar long-standing services in the system). The system uses a Support Vector Regression (SVR-) model [20, 21] which is largely employed for forecasting data in both linear and nonlinear problems. The system trains the SVR-model using the normalized QoS and reputation values of all services in the system. Initial QoS values of newcomer services are then used by the SVR-model to estimate their initial reputation values.

The main goal of this work is two-fold:

– First, it provides a solution to evaluate the reputation of newcomer web services that correctly reflects their quality. This enhances their visibility to end users and improve the overall quality of the recommendation system, by providing a solution to the cold-start problem.

– Second, it provides a solution to the whitewashing problem by looking to functionally similar services (including services that have left the system) which are recorded in the system registries.

We evaluated the proposed reputation estimation method on a set of real-world web services. We compared the obtained results with competing approaches from the state of the art.

The remaining of the paper is organized as follows. Section 2 presents the related work. Section 3, provides the reputation estimation algorithm. Phases one, two and three of the method are detailed respectively in Sections 4, 5, and 6. Section 7 discusses results of the experiment and Section 8 concludes the paper.

2. Related Work

Reputation management has been successfully studied in different computer science domains such as in e-commerce and online information systems (e.g. [22–26]), multi-agent systems (e.g. [27–30]), peer to peer (P2P) and social networks (e.g. [31–33]), and mobile and ad-hoc networking (e.g. [34–36]).

Leveraging reputation have being also explored in Network Functions Virtualization (NFV), Software-Defined Network (SDN), Fog and cloud computing technologies that enhance the flexibility of network function provision and update. Security, reputation and trust become crucial issues in practical deployment of the aforementioned technologies [37–41]. More precisely, reputation is proposed as a mechanism to control malicious SDN Applications, SDN Controllers and VNFs use cases (i.e. NFVI-as-a-service, VNF-as-a-service, Virtual-network-platform-as-a-service, etc.) For instance, Dynamic and multidimensional trust and reputation access controls mechanisms have been suggested in [42]. Challenges of incorporating trust in FNV are discussed in [43]. Authors in [44] highlighted the set of possible attacks in a cloud environment as they had provided a reflection on the need for a trusted NFV and management and orchestration components. In [37], the authors identify the problem of the lack of trust architecture for NFV and they provide the necessary requirements for establishing trust in it.

Authors in [45] proposed a reputation-based scheme to identify rogue/-malicious controllers in a distributed environment. The scheme is based on trust and reputation which is centrally managed. Authors in [46] proposed a collaborative scheme formed amongst SDN domains in the path of attacks.

The scheme is capable of cutting off malicious flows via (i) reputation-based cooperation amongst SDN domains that may be managed as disparate Autonomous Systems, and (ii) distribution of the mitigation process through transit SDN domains, aiming at dropping malicious flows near their origin.

The subject is still challenging and important in this field and the most of the established trust and reputation models lack of providing rigorous methods to estimate the initial reputation/trust values of SDN/FNV services.

For cloud and service-oriented computing systems, many reputation management methods have been proposed (e.g. [1–7, 12, 47–50]). However, little efforts have been dedicated to the study of newcomer reputation estimation.

In fact, most of the proposed reputation approaches do not consider thoroughly this aspect, and a few of them provide default bootstrapping techniques [15, 16, 51–53], i.e. they assign a default constant reputation score, such as a low (0), a neutral (0.5), a high (the maximum) or no reputation value, to all newcomers. For instance, Zacharia *et al.* [51] give the minimum possible trust value, and [53] assigns a neutral value (0.5). However, in such situation, newcomers may never have a chance to get selected. Even in the case of assigning high trust values, the problem of "whitewashers" (i.e. malicious participants that leave the system and come back with new identities) could raise.

The framework proposed by Jin-Dian *et al.* [54] assigns the provider reputation to its newly posted services. The authors suggest assessing the reputation of the provider based on its past experiences. However, the problem appears if the provider is a newcomer in the recommendation system.

Feldman and Chuang [17] propose a solution for bootstrapping the reputation of newcomer services based on its probability of deceiving. This probability is computed by collecting all transaction information of the newcomer's first-time interaction. This approach is community-based, and newcomer reputation is adjusted to the reputation of others. However, initial reputation scores are still not fair and they do not reflect the actual reputation of newcomers.

Malik and Bouguettaya [11] propose two bootstrapping techniques for establishing the reputation of newcomer web services. The first is an adaptive technique that assigns the initial reputation value based on the rate of maliciousness in the system. The second approach assigns a default reputation score to a newcomer service, where the initial reputation is purchased from the community provider. Or, the community requests some evaluators (elder service with high reputation) to evaluate the newcomer service in a short

period of time. In the first technique, the reputation of a specific web service is related to the maliciousness rate in the community, which seems penalizing or rewarding based on a factor that is unrelated to the service itself. In the second technique, the contribution and the impact of requesters on the reputation of web services are very high, which raises the problem of the trust of evaluators themselves.

Huang *et al.* [10] propose an equitable trustworthy mechanism that enables new services to startup and grow in an ecosystem environment. The mechanism distinguishes between novice and mature services during service recommendation. The approach considers two trust bootstrapping strategies: i) default strategy where they assign to the newcomer a default initial trust value, and ii) an adaptive bootstrapping strategy where they assign to the newcomer the average trust value in the system. The first strategy does not provide a solution for the cold-start problem and for the whitewashing problem in the case of assigning a high value. Moreover, the second technique assigns the average trust in the system to newcomer services, which is not always an accurate solution (e.g. the case where the average is high and the service is bad or the inverse).

Wu *et al.* [15] introduce a neural network based approach for bootstrapping the reputation of web services. The approach builds a model that learns possible correlations between features and performances of existing services using Artificial Neural Networks. Then, it generalizes findings to establish tentative reputation values when it evaluates new and unknown services. This approach depends on features that are gathered from service providers by filling a specific form without taking into consideration the whitewashing cases.

The main differences between the proposed solution and the previous solutions can be summarized as follows. First, instead of assigning the same reputation value to all newcomer web services, we propose to estimate the initial reputation value of a newcomer service based on its provider's reputation, reputation values of its similar long-standing services, and its initial QoS values. When all the previous values are not present, we make an estimation using a Support vector regression model built from the reputation and QoS values of long-standing services. In addition, we propose a solution to overcome the whitewashing problem based on similarity of the newcomer service with registered services that have left the system.

Input: S_i // Newcomer service
Output: $\hat{R}_i$ // Estimated Reputation
Begin

1: $\lambda = 0.3$; TopkBool = false;
2: **if** (Provider(S_i) $\in$ *ProviderList*) **then**
3: prReputation = providerReputation(Provider(S_i)) ; // compute providers reputation
4: simServiceSet = getSimilarServices(S_i,*ServiceList*); // get functionnaly similar services
5: **if** (simServiceSet$\neq \emptyset$) **then**
6: **for all** ($S_j \in$ simServiceSet) **do**
7: QoSim [S_j.index] = ρ(normalize(Qos(S_i)),normalize(Qos(S_j))); // using Eq. 4
8: **if** (QoSim [S_j.index] > 0) **then**
9: TopKset.add(S_j) ; // TopK neighbors selection
10: **end if**
11: **end for**
12: **if** (TopKset ==$\emptyset$) **then**
13: TopkBool = true ; // To continue from line 33
14: **else**
15: R_{Min} = MinReputation(TopKset) ;
16: R_{Max} = MaxReputation(TopKset) ;
17: **if** ($R_{Max} - R_{Min} < \lambda$) **then**
18: num = denom = 0 ; // Technique 1, Using Eq. 6
19: **for all** ($S_j \in$ TopKset) **do**
20: num += QoSim [S_j.index] $\times$ getReputation(S_j) ;
21: denom += QoSim [S_j.index] ;
22: **end for**
23: $\hat{R}_i = \frac{num}{denom}$;
24: **else**
25: MLRmodel =buildMLRegressionModel(topKset) ; //Tech. 2 - linear regression
26: $\hat{R}_i$= estimateReputation(Qos(S_i),MLRmodel) ;
27: **end if**
28: **end if**
29: **else**
30: $\hat{R}_i$= prReputation ; // Assign provider reputation
31: **end if**
32: **end if**
33: **if** (!(Provider(S_i) $\in$ *ProviderList*) || TopkBool) **then**
34: simServiceSet = getSimilarServices(S_i,*ServiceList*);
35: **if** (simServiceSet$\neq \emptyset$) **then**
36: SimVector = Similarities(S_i, simServiceSet);
37: aService =HighestScoreService(SimVector);
38: **if** (Max(simVector)==1 && hasLeft(aService)) **then**
39: $\hat{R}_i$ =getReputation(aService);
40: **else**
41: MLRmodel =buildMLRegressionModel(simServiceSet) ; //Apply tech.2 for SimServSet
42: $\hat{R}_i$= estimateReputation(Qos(S_i),MLRmodel) ;
43: **end if**
44: **else**
45: SVRModel = BuildSVRModel(serviceList); // Build Support Vector Regression model
46: $\hat{R}_i$ = estimateSVRReputation(Qos(S_i),SVRModel);
47: **end if**
48: **end if**
End

Algorithm 1: Reputation estimation algorithm

3. Reputation estimation method

We suppose that we have a Service Recommendation System (SRS) that offers useful suggestions to its clients, helping them in the selection of appropriate services that fulfill their business needs. The SRS recommends services based on their QoS and reputation values. The architecture of the SRS is composed from many modules that are responsible for:

1. Registering service providers and their services.
2. Registering clients.
3. Storing and updating service related QoS and information. Eventually, with a monitoring component, or by allowing providers to update their service QoS.
4. Collecting feedback rating from clients.
5. Evaluating reputation values of services from their user feedback ratings.
6. Indexing/Archiving service description, QoS, reputation, and information for services and providers including those which tend to leave the system.
7. Recommending services using Recommendation algorithms based on QoS and Reputation.
8. Bootstrapping newcomers reputation; i.e. estimating the reputation of new services in the System.

In this work, we focus only on the estimation of newcomer services (point 8). Other modules are supposed implemented and working properly.

When a newcomer service S_i arrives to the system, we assume that it comes with an initial QoS vector $Q_{S_i}^{init} =< q_{i,1}, q_{i,2}, ..., q_{i,k} >$. These QoS values ($Q_{i,j}$) are provided during registration time by the service provider ($Pr(S_i)$) as advertised QoS data. These data can also be updated by the system after a period of service monitoring and testing (for that propose, the system can use one of the approaches proposed in this survey [55].)

Algorithm 1 presents the process that covers three phases for estimating the reputation of a newcomer web service S_i. First, the system checks whether the service provider is recorded by the system, that is, the service provider belongs to the list of providers *ProviderList* that have published services in the system. In the positive case, the system calculates the reputation of this provider, denoted *prReputation*, based on the reputation values

of its long-standing services (Line 3 in Algorithm 1). Section 4 gives more details on how provider's reputation is calculated.

Afterwards, the system looks for long-standing services which provide similar functionalities to those provided by the newcomer service (Line 4). To evaluate the functional similarity between services, we use the approach proposed in [56]. If *simServiceSet*, which denotes the set of similar service, is not empty, the system selects, using positive Spearman's Coefficient values (ρ), a subset of top-K neighbors with close QoS vectors to the newcomer's QoS vector (Lines 6-11). Both Spearman's coefficient and top-K are effectively used in recommender systems for the selection of similar elements[57].

When the maximum distance between reputation values of neighbor services in Top-K is relatively small (less than $\lambda = 0.3$ for instance), which means all reputation values of neighbors are close to each other, the system estimates the reputation of the newcomer service as the mean weighted reputation values of its neighbors, where weights are their Spearman's coefficient values (Lines 15-23). Otherwise, the system builds a multiple regression model (see Section 5.4.2 for details) using QoS vectors and reputation values of top-K neighbors, and therefore estimates the reputation value of S_i using this model (Lines 25-26).

In the case where *simServiceSet* is empty, the system assigns to the reputation of S_i the reputation values of its provider "prProvider" (Line 30). This is motivated by the fact that if the provider has a good reputation, it is likely that its new web service will have a good starting reputation too.

Besides, when the provider of the service is also new in the system, the system checks if it is a whitewashing situation (lines 33-38 in Algorithm 1). The system retrieves all similar long-standing services, including archived service, and then compares their similarity scores with the newcomer service (similarity scores range between 0 and 1, where 1 means that services are totally similar and 0 otherwise). If the highest similarity score equates to one, and the similar service has left the system, then, the provider of the newcomer service becomes suspicious, and we assign the (old) reputation value of the left service to the reputation values of the newcomer service (line 38 in the algorithm). Otherwise, we use Technique 2, where the system builds a multiple linear regression model from QoS and reputation values of similar services ($simServiceSet$). The system estimates the reputation of S_i using this model (Lines 41-42).

When the newcomer service and its provider are both new, and there is no similar services in the system, we go to Phase 3 (detailed in Section 6).

The system builds a Support Vector Regression model from QoS vectors and reputation values of all long-standing services in the system (line 44 in the algorithm). Similarly to Phase 2, the model gives also an estimation of service reputation based on service's initial QoS. The estimated value is assigned to the reputation of the newcomer service S_i.

4. Provider reputation

The reputation of a provider mainly depends on the quality of its offered services, thus on their reputation values. In this phase, we calculate the reputation of a given provider as the weighted arithmetic mean of reputation values of its services. Given a provider Pr_x, let $Services(Pr_x) = \{S_i\}, i = 1, ..., n$ denote the set of n services provided by Pr_x, and $\Omega(S_i)$ be the number of users who rated service S_i. The reputation of a provider Pr_x is calculated as follows:

$$RP(Pr_x) = \begin{cases} \frac{(\sum_{i=1}^{n} \Omega(S_i) * R_i)}{\sum_{i=1}^{n} \Omega(S_i)} & if \quad Services(Pr_x) \neq \emptyset \\ 0 & Otherwise \end{cases} \tag{1}$$

where,

- R_i is the reputation of service S_i that belongs to the provider's service set ($S_i \in services(Pr_x)$).
- $\emptyset$ denotes the empty set.

The new providers reputation when it introduces its first service, is equivalent to its service reputation that the system estimates using one of the following estimation phases.

5. Reputation estimation from similar services

Since users rate functionally-similar web services (i.e. services that provides same functionalities) based on the same criteria, we consider that it is possible to estimate the reputation of newcomer web services using reputation scores of its similar services, which are calculated by aggregating users' ratings. In this phase, a four-step technique is proposed to estimate this reputation based on QoS values of the newcomer service S_i and the existing long-standing similar services (see Figure 1). As mentioned above, the initial QoS values of S_i are represented by the QoS vector $Q_{S_i}^{init} =< q_{i,1}, q_{i,2}, ..., q_{i,k} >$.

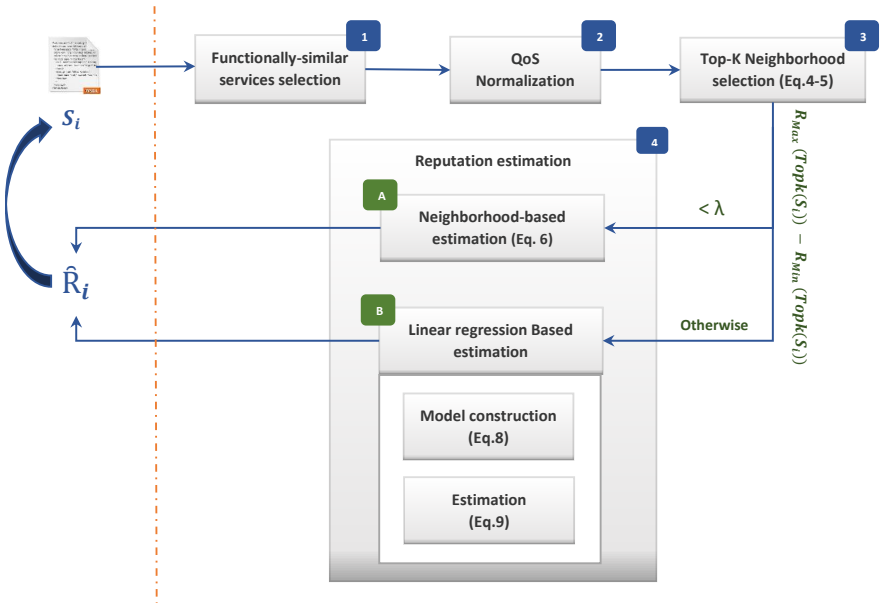

Figure 1: The second phase for estimating the reputation of a newcomer service S_i.

The method for estimating the initial reputation of S_i denoted $\hat{R}_i$ is detailed in the following sub-sections.

5.1. Step 1 - Functionally-similar service selection

First, the system selects from its databases long-standing services that offer the same functionalities (e.g. Weather forecasting services, currency services, transportation services, etc.) to the service S_i. In the literature many approaches that compute the similarity between web services are proposed (e.g. [58–60]). In this work, we choose to use the approach proposed in [60] to calculate the similarity between a newcomer web services and the existing services. The approach assesses the similarity between two web services by comparing their WSDL definitions using several lexical and semantic metrics. The similarity value is a score that indicates to what level two compared services are close to each other (i.e. how much these services are similar). Similarity value could range between 0 and 1, where 0 means that the compared services are totally different, and 1 means that these services are totally similar. The similarity threshold is a starting value to consider two compared services as similar. In fact, five classes of similarity can be defined, (very high, high, medium, low, and very low). A similarity value

that ranges in [0.6, 0.8[is considered as a high similarity value, while a value that ranges in [0, 0.2[is a very low similarity value. In this work, we have fixed the similarity thresholds to 0.75 (based on [60] experiments), which means that the system accepts services with high similarity values to ensure that these services are offering close functionalities to the newcomer one. We recommend thresholds be-longing to the first two classes (High and very high). The selection of more relaxed threshold values leads to the selection of a larger set of services, and hence a larger bias between their QoS (and reputation) occurs. This influences negatively the accuracy of estimating the new-comers reputation. The result of this step is a set of similar services denoted by $simServiceSet$.

5.2. Step 2 - QoS normalization

Second, the system retrieves and normalizes QoS and reputation values of each service in the $simServiceSet$. Let $simServiceSet = \{S_j\}, (j = 1, .., m)$ be the set of m similar services to the newcomer service S_i. Each similar service S_j in this set has a QoS vector $Q_{S_j} =< q_{j,1}, q_{j,2}, ..., q_{j,k} >$ and a reputation value R_j calculated by the system from user feedback ratings. Besides, the newcomer service S_i is defined by the vector $Q_{S_i}^{init}$ that represents its initial known QoS values, and $\hat{R}_i$ that represents the unknown reputation value (to be estimated).

Afterwards, the system normalizes all QoS values in the range $[0, 1]$.

Thus, each QoS value, $QosVal \in \{q_{j,l}, (j = 1, .., m; \quad l = 1, .., k)\}$ is replaced in its vector by its normalized value $NewQosVal$ which is calculated as follows:

$$NewQosVal = \frac{QosVal - MinVal}{MaxVal - MinVal} \tag{2}$$

Where $MinVal$ and $MaxVal$ are respectively the minimum and maximum recorded values in the system for that QoS metric (with the index l). Note that some of QoS metrics have values that are interpreted inversely, i.e. the higher is the value, the lower is the quality. This includes execution time and price. Thus, the scaled value $NewQosVal$ is calculated as follows:

$$NewQosVal = 1 - (\frac{QosVal - MinVal}{MaxVal - MinVal}) \tag{3}$$

5.3. Step 3 - QoS-similar neighborhood selection

The more the QoS values of the newcomer services are close to the QoS values of other services, the more its reputation value is close to their reputation values. Thus, the system in the third step selects the QoS-Similar

neighbors from the *simServiceSet* by calculating similarities between the QoS Vectors of Web services. These similarities could be calculated using PCC (Pearson Correlation Coefficient), Spearman's rank correlation coefficient (The Spearman coefficient in short) or VSS (Vector Space Model Similarity) estimates, which are used in recommendation systems [57, 61–63]. PCC can generally achieve higher performance than VSS [64] and Spearman coefficient achieves more reliable results in finding QoS-based similar services Than PCC [57]. Therefore, we employ ρ (Spearmanś rank correlation coefficient) for the similarity computation between normalized QoS vectors $Q_{S_j}, (j = 1..m)$ and $Q_{S_i}^{init}$, the QoS vector of the newcomer S_i.

The Spearman correlation evaluates the monotonic relationship between two continuous or ordinal variables. In a monotonic relationship, the variables tend to change together, but not necessarily at a constant rate. The Spearman correlation coefficient is based on the ranked values for each variable (Rank of the QoS among others in the same vector) rather than the raw data (QoS values themselves). This coefficient calculates the similarity between two service vectors by considering the difference of the two rankings for each quality in the vectors. Spearman coefficient is calculated as follows (Equation 4):

$$\rho(S_i, S_j) = 1 - \frac{6 \sum d^2}{m(m^2 - 1)} \tag{4}$$

where, m is the number of qualities in each vector, and d is the difference of the two ranking for each quality (i.e. for each item in the vector).

From Eq. 4, $\rho(S_i, S_j)$ values belong to the interval $[-1, 1]$, where a larger ρ value indicates higher QoS-similarity between services S_i and S_j. After calculating QoS-Similarities between the newcomer service and services in the $simServiceSet$, a set of top-K neighbors is identified based on ρ values. In this work we ignore negative ρ values because negative values could represent a dissimilarity between compared services, which influences greatly on the accuracy of the estimation of reputation in next steps. Thus, the top-K neighbor set of the newcomer service S_i is defined as follows:

$$\begin{aligned} TopK(S_i) = \{S_j \quad | \quad & \rho(S_i, S_j) >> 0; \\ & S_j \in simServiceSet\} \end{aligned} \tag{5}$$

where, $\rho(S_i, S_j)$ is computed using Eq. 4. In case $TopK(S_i)$ equates the empty set ($\emptyset$), then the system moves to Phase 3 of the method, and the

estimation of the reputation of the newcomer service $\hat{R}_i$ is calculated using the SVR-based model. Otherwise, its reputation is estimated in the next step.

5.4. Step 4 - Reputation estimation

We propose two techniques to estimate the reputation of a newcomer service based on the data about the services in the top-K neighbor set ($TopK(S_i)$). The first consists in calculating the weighted mean of neighbors reputations (Section 5.4.1), and the second is based on the construction of a multiple linear regression model (Section 5.4.2) from QoS vectors and their corresponding reputation values.

The system uses the first technique when the difference between the maximum and the minimum reputation values of services in the Top-k set is less or equal than a threshold λ (e.g. $\lambda = 0.3$).

$$R_{Max}(TopK(S_i)) - R_{Min}(TopK(S_i)) < \lambda$$

In fact, Reputation is a value that ranges between 0 and 1. The higher is the value, the more trusted is the service. λ is the upper boundary of acceptable variation between reputation values in the Top-K neighbor set. A deviation up to $\lambda = 0.3$ between reputation of two elements is considered natural based on variation of user preferences (i.e., two honest persons can give different but close rates to the same service). We selected the upper boundary ($\lambda = 0.3$) to apply the neighbor based estimation because these neighbors are similar to the newcomer services and they have close QoS. However, if more relaxed boundaries are selected, then the difference between neighbors reputation values is larger, and hence attributing the mean of these values may result in a significant bias when estimating newcomers reputation.

In case the difference between the minimum and maximum reputation in the top-k neighbor set is greater than λ, the system selects the second technique to estimate $\hat{R}_i$.

5.4.1. Neighborhood-based estimation

The estimation of $\hat{R}_i$ is calculated using Eq. 6.

$$\hat{R}_i = \frac{\sum_{j \in TopK(S_i)} (\rho(S_i, S_j) * R_j)}{\sum_{j \in TopK(S_i)} \rho(S_i, S_j)} \tag{6}$$

where $TopK(S_i)$ is the set of neighbors that are functionally and qualitatively (based on their QoS values) similar to the newcomer service S_i, and $\rho(S_i, S_j)$

is the similarity between S_i's and S_j's QoS vectors. Using ρ as a weight in Eq. 6 means that reputation scores of services, whose QoS values are highly correlated with the QoS values of the newcomer service, are assigned with higher weights (i.e ρ values close to 1).

5.4.2. Linear regression-based estimation

The second technique to estimate $\hat{R}_i$ is achieved by constructing a multiple regression model, using QoS and reputation values of top-K service neighbors.

Multiple regressions are statistical techniques used for predicting unknown Y values (a dependent variable) corresponding to a set of X values (independent variables). In our study, the multiple regression is expected to give a model that could relate the reputation values of long-standing services to their QoS values, that is, we consider the dependent variable Y to represent the reputation of services as a function of multiple QoS attributes (independent variables) such as *response time, availability, throughput, latency, price, etc.* Thus, if we have m services in the $TopK(S_i)$ (S_j , $j = 1, 2, ..., m$), and each service S_j has a QoS vector $Q_{S_j} =< q_{j,1}, q_{j,2}, ..., q_{j,k} >$ that holds k QoS metrics, and each service S_j has a reputation value $R(S_j)$ denoted R_j, the relationships between reputation (the dependent variable) and QoS metrics (independent variables) can be expressed by the following equation:

$$\underbrace{\begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,k} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m,1} & q_{m,2} & \cdots & q_{m,k} \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}}_{\beta} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}}_{\varepsilon} = \underbrace{\begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{pmatrix}}_{Y} \tag{7}$$

where :

- X is the design matrix that packs all regressors (predictors) $q_{l,j}, l = 1, ..., m$ and $j = 1, ..., k$.
- β is the regression coefficient vector (called also slop vector).
- ε is the error vector. Error terms $\epsilon_l, i = 1, .., m$ capture all the factors which influence the dependent variable ($R_l, l = 1, ..., m$) other than regressors ($X_{l,j}, l = 1, ..., m$ and $j = 1, ..., k$).

The multiple regression of the model can be simplified to:

$$R_l = \beta_1 q_{l,1} + \beta_2 q_{l,2} + ... + \beta_k q_{l,k} + \epsilon_l, \qquad l = 1, ..., m \tag{8}$$

where,

- R_l is the response (estimated reputation) of the linear combination of the model terms.
- β_l $(l = 1, ..., k)$ represents the unknown coefficients.
- ϵ_l is the error term.

After model construction, the system uses solved values of the unknown coefficients (β_l $(l = 1, ..., k)$, and the error term (ε)), to estimate the reputation of the newcomer service based on its initial QoS vector using Eq 9.

$$\hat{R}_i = \beta_1 q_{i,1} + \beta_2 q_{i,2} + ... + \beta_k q_{i,k} + \epsilon_i. \tag{9}$$

6. SVR-based reputation estimation

In this phase, the system deals with general cases, where it is unable to recognize the service provider but not similar services. This phase relies on the use of a Support Vector Regression model [20, 21] to estimate the reputation of newcomer services. The system trains an SVR model using QoS data and reputation values of all long-standing services. The output model is used by the system for reputation estimation. The essence of the application of this algorithm in our context is to map QoS and reputation values of long-standing services into a higher dimensional space via a non-linear mapping using RBF Kernel, and then to do linear regression in this space. The definition and motivation of using SVR are presenting bellow.

6.1. Formal definition

The main idea in an SVR can be formalized as a problem of inferring a function $y = f(x)$ based on the training set set $D = (x_i, y_i); i = 1, 2, ..., N$, $y_i \in \mathbb{R}$, $x_i \in \mathbb{R}^N$, where x_i is the ith input in the N-dimension space, and y_i is the output value corresponding to x_i. Furthermore, learning an SVR is equivalent to finding a regression function of the form of Eq.10 :

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k(x, x_i) + b \tag{10}$$

where :

- $k(x, x_i) = \Phi(x).\Phi(x_i)$ is a kernel function that maps input data x_i to a high dimension feature space that can describe the relationships between inputs x_i and outputs y_i.
- $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)^T$ and $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_N^*)^T$ and b are the parameter of the model.

The parameters α_i and $\alpha_i^*(i = 1, ..., N)$ can be calculated by minimizing the following objective function (Eq.11)

$$\frac{1}{2}\sum_{i,j=1}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) + \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) - y\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) \quad (11)$$

which is subject to the following constraint (Eq.12):

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \quad and \quad \alpha_i, \alpha_i^* \in [0, C] \quad (12)$$

Different SVR models can be elaborated by selecting different kernel functions. In this work, we use the Gaussian Radial Basis Function (RBF) which performs a nonlinear mapping between the input space and a high dimensional space, and which is easy to implement at the same time [65]. Under the assumption that reputation is non-linear with QoS data, we choose the RBF Kernel to develop our reputation estimation model. The model can be rewritten as follows:

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\exp\left(\frac{-\|x_i - x\|^2}{2\sigma^2}\right) + b \quad (13)$$

where σ, the kernel parameter, is the width of the RBF kernel function. x_i is the input vector of the training data, i.e. the QoS vector of long-standing services in our case. x is the vector of testing data, i.e. the initial QoS vector of the newcomer service. The reputation of the newcomer service $\hat{R}_i$ equates to f(x).

6.2. Motivation for using SVR

Theoretically, SVR has many advantages over other regression and machine learning techniques due to many technical characteristics well-know in the literature [20]. Practically, SVR has been proven to be an effective tool in real-value function estimation (e.g. [65–70]), as it has been shown experimentally by evaluating its performance over other machine learning technique including Neural networks, multiple regression, LASSO regression, Naradaya-Watson Kernel estimators, K-NN, regression tree, boosting, bagging or random forests, etc. All these advantages strongly motivated us to apply this technique on the estimation of newcomers' reputation, exploiting the relationship between QoS and reputation of other services.

7. Experiments and Evaluation

To evaluate the proposed reputation estimation method, we conducted an experiment on a set of real web services collected from WSDream [71] and QWS [72] datasets. Our experiment has been conducted through three steps, in which we evaluated the three proposed estimation phases. Different accuracy metrics are used to compare the results obtained by our estimation method against results obtained by related state-of-the-art methods.

7.1. Data description and preparation

WSDream dataset holds 5825 web service QoS data evaluated by 339 users in different geographical locations (we have chosen the dataset 2 in WSDream). This dataset holds 339*5825*2 (2 QoS characteristics: response time and throughput). QWS dataset holds 365 web services with 9 QoS characteristics listed in Table 1.

To use a maximum number of QoS metrics with different monitored values of response time and throughput, we selected the services that belong to the two datasets. We matched web services based on their URIs, Names, and WSDL file size. We obtained 409 WSDL files for 356 services, where 53 files from the 409 are redundant WSDL with different endpoint and QoS metrics. Each service in this set has 7 fixed QoS metrics from QWS, and 2 QoS metrics (response time and throughput) that vary based on the observation of 339 users from WSDream.

It is important to note that we consider QoS from QWS as the providers advertised QoS. And, we use the two other QoS metrics (response time and throughput) from WSDream when simulating user feedback ratings for the

Number	Quality	Description	Unit
1	Response time	Time taken to send a request and receive a response	ms
2	Availability	Number of successful invocations / total invocations	%
3	Throughput	Total Number of invocations for a given period of time	invocations/second
4	Successability	Number of response messages / number of request messages	%
5	Reliability	Number of error messages / total number of messages	%
6	Compliance	The extent to which a WSDL document follows WSDL specification	%
7	Best Practices	The extent to which a Web service follows WS-I Basic Profile	%
8	Latency	Time taken for the server to process a given request	ms
9	Documentation	Measure of documentation (i.e. description tags) in WSDL	%

Table 1: QoS metrics selected from QWS dataset

long-standing services, considering by the variation between these QoS different perceptions on these services. Generally, there is always a variation in the delivered QoS based on location, networking overload, server status, etc. Hence, different feedback ratings come from users based on these perceptions. This deviation between users is natural and we think that it helps in collecting correct feedbacks from users viewpoint.

7.2. Evaluation metrics

Statistical accuracy metrics are performance metrics that are used for the evaluation of recommender systems. In this experiment, we use Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root-Mean-Squared-Error (RMSE) metrics to measure the quality of the estimation provided by our method in comparison with similar methods.

MAE is a quantity that measures how close are the estimations (predictions) to the eventual outcomes. MAE is defined as follows:

$$MAE = \frac{\sum_{i=1}^{n} |R_i - \hat{R}_i|}{n} \tag{14}$$

MAPE expresses the accuracy as a percentage of the error (e.g. if MAPE is 5, the estimation is off by 5 %). MAPE is defined as follows:

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{R_i - \hat{R}_i}{R_i} \right| \times 100 \tag{15}$$

RMSE is a quantity to measure the difference between predictions and eventual outcomes. RMSE gives a relatively high weight to larger errors. It

is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{R}_i - R_i)^2}{n}} \tag{16}$$

In Eqs. 14 and 16, R_i denotes the actual reputation (Reputation which is calculated by aggregating simulated feedback ratings), and $\hat{R}_i$ denotes the estimated (predicted) reputation calculated by the proposed method (or similar methods), and n is the number of tested services.

In addition, we use the correlation coefficient (R) that measures the strength and the direction of a linear relationship between variables (Reputaion and Qos metrics in this case), and the coefficient of determination (R^2) that gives the proportion of the variance of reputation variable predictable from QoS variables.

7.3. Comparison

To show the efficiency of the proposed method, we compare our Reputation Estimation Method (labeled **REM**) with the following two competing methods (1 and 2) and the three baseline methods (3 to 5):

1. The method based on Malicious user Density (labeled **MDBM**) proposed by Malik et al. [11]: this adaptive bootstrapping approach calculates the initial trust value based on the rate of maliciousness in the system. It assigns a high initial reputation value when R, the maliciousness rate, is low, and a low reputation value when R is high.
2. "Artificial Neural Network"-based method (labeled **ANN**) proposed by Wu *et al.* [15]. This machine learning method estimates the reputation of newcomer Web services using an artificial neural network model built from Quality of Service values.
3. Minimum Value Method (labeled **MVM**): this approach is used by Zacharia et al. [51], it assigns the minimum possible reputation value to all newcomers.
4. Neutral Method (labeled **NM**) used by Wang et al. [53]. This method assigns the neutral value (0.5) to the reputation of newcomers.
5. Average Reputation Method (labeled **ARM**) used by Huang et al. [10]. This method assigns the average reputation in the system (i.e. the mean reputation value of longstanding services)

7.4. Feedback rating simulation

Unfortunately, the unavailability of reputation datasets for web services drives many researchers (e.g. [11, 19, 58, 73, 74]) to simulate feedback rating. The automated rating process provides feedback that corresponds to the level of satisfaction/dissatisfaction with service quality [74]. Fortunately, this is the essence of the expectancy-disconfirmation theory from market science [75], which provides a conceptual framework for the study of consumer satisfaction versus service quality. According to this theory, consumer satisfaction is the outcome of the comparison between consumers preconsumption expectation and postconsumption disconfirmation, where confirmed expectations lead to moderate satisfaction, positively disconfirmed (i.e., exceeded) expectations lead to high satisfaction, and negatively disconfirmed (i.e., underachieved) expectations affect satisfaction more strongly than positive disconfirmation and lead to dissatisfaction. User expectations in our context are the quality (QoS) ensured by the service. User satisfaction, which is manifested by her/his feedback ratings, is hence depending on service QoS based on the expectancy-disconfirmation theory.

The generation/simulation of reputation scores considers the assumption that there is a correlation between QoS and reputation scores. Thereby the goal of this experimentation is not to show that the proposed model gives results that are good regarding this assumption. The goal is however to evaluate the influence of maliciousness in the estimation of reputation scores and the comparison of the proposed model with existing models from the literature, in presence of the aforementioned assumption.

Likewise, we have built a Java program that simulates interactions between a set of 409 web services and a set of 339 users. Each service has an actual performance level (i.e., overall quality), denoted by *PerfVal*. This performance level represent how good is the overall quality provided by the service on a scale of 10. PerfVal is calculated based on a utility function, i.e., a single scalar metric to quantify quality perception of the delivered service, as suggested in [74]. However, in our work, we propose to calculate the utility function with the root mean square, which is a measure of the magnitude of the scaled QoS metrics.

Thus, *PerfVal* of service S_i is assessed as follows:

$$PerfVal(S_i) = 10 \times \sqrt{\frac{\sum_{j=1}^{k} Scal(Q_{i,j})^2}{k}} \tag{17}$$

where, k is the number of used QoS metrics (Q_j, $j = 1, .., k$). And, $Scal(Q_{i,j})$ is the scaling function, which is defined by Eq. 18, if the quality is positive (i.e., the higher is the value the higher is the quality), and by 1 - the same formula otherwise.

$$Scal(Q_{i,j}) = \frac{Q_{i,j} - Min(Q_j)}{Max(Q_i) - Min(Q_j)} \tag{18}$$

$Min(Q_j)$ and $Max(Q_j)$ are respectively the minimum and maximum recorded values of the quality Q_j.

The program simulates two kinds of users: honest and malicious users. Honest users randomly rate a service based on its *PerfVal* within the interval $[Max(0, PerfVal-2), Min(PerfVal+2, 10)]$. For example, if *PerfVal*=7, fair feedback ratings could be 5, 6, 7, 8, and 9. The deviation with ± 2 from *PerfVal* represents a natural variation between user opinions.

Three types of malicious users with different behaviors are simulated:

1. Pure malicious users that gives unfair feedback rating in all interactions (i.e., rates outsides the expected interval).
2. Gray users that start by providing fair feedback ratings in half of the interactions and then provides unfair ratings in the remaining interactions.
3. Malicious users with an oscillating behavior, which provide fair and unfair ratings in an oscillating manner.

The malicious density used in each run of the experiment holds one third (1/3) from each type of malicious users. Even though Whitby et al. [76] and Malik et al. [19] claim that high maliciousness densities are unrealistic in real world applications, we variating maliciousness densities in the interval [10%-70%] to analyze how well the model performs in such conditions and consequently could drive a safe conclusion about the models performances. In this simulation, we are considering up to 339 reviews for each service in the experiment. Obtained data are gathered from 10 execution runs. The results are the mean of these 10-round data

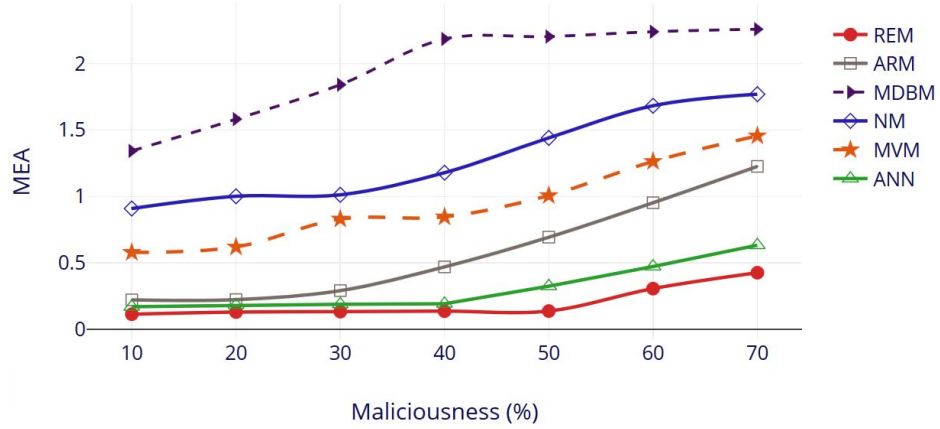

Figure 2: MEA results of our provider-based estimation and other methods with variation of maliciousness rates.

7.5. Provider-based reputation

In this section, we evaluate the accuracy of the reputation estimation of newcomers using the reputation of their providers. We have selected a list of providers that have more than one service in the working dataset. For each provider, the program takes randomly one or more services, depending on the number of its published services and the rate of test, to construct the test-set (i.e. the set of services that are considered as newcomers to the system). The program does the following:

1. Calculate the reputation from simulated feedbacks for all services.
2. Randomly select services to construct the test-set based on a given ratio. This ratio indicates how many services will be considered as a newcomer. Note that each service in the test-set has a calculated reputation.
3. For each element in the test-set, calculate its provider reputation and assign it to the newcomer's estimated reputation.
4. Evaluate the reputation of the tested service using other methods.
5. For each method, compute the MAE, MAPE and MRSE based on the calculated (from simulated feedbacks) and the estimated reputation (evaluated by the method).
6. Print results.

7. Repeat steps (1-6) for different test-set rates.

The program identified 55 providers that publish more than one service from a total of 251 providers in the working dataset.

We have performed several runs, varying the maliciousness density (i.e, the number of malicious users) from 10% to 70%, and using a test set of 30% of services. Figure 2 shows the obtained MAE for our method (**REM**) and other methods. Similar results are obtained with MAPE and RMSE. As we can see, the MAE values given by our method are very close to 0, when malicious density varies between 10% and 50%, which means that there is a very slight variation between the calculated reputation from simulation and the estimated reputation from newcomers' providers. Starting from 50% to 70% of malicious users, the MEA obtained by our method increases. However, the estimation of newcomers is still acceptable and better than all other methods. Both, the Artificial Neural Network based method (**ANN**) and the assignment of the average reputation in the system (**ARM**) to newcomers, give a good reputation estimation when maliciousness density varies up to 30%. ANN keeps giving good reputation estimation represented by lower MEA values. The other methods are influenced by the maliciousness rates in the system. Even that REM gives better reputation estimation results, It is obvious that the system could not apply this method, especially when the provider of the newcomer service is new to the system too.

7.6. Similarity-based reputation

This section presents the results of the experiment conducted to evaluate the performance of Phase 2: similarity-based reputation estimation.

Only services that have similar services are evaluated during this phase. Hence, the program eliminates from the working dataset, all services that have none, one or two similar services. That is, we need at least 3 services (i.e. one for test and two at least to apply Eq. 6) to evaluate one of them using Technique 1 (Section 5.4.1). We need at least 10 similar services (i.e. one service for a test, and at least 9 services data to build the linear regression model with 9 known QoS attributes) to evaluate newcomer reputation using Technique 2.

First, the program analyzed the initial service dataset. It identified 17 groups (clusters) of services with 173 services, as it is shown in Table 2. Elements of each group are functionally similar to each other. Column "Nbr

Group	Nbr Services	Nbr Tests	MAE	MAPE%	RMSE
Booking	3	1	0.032	0.504	0.045
Calendar	11	4	0.036	0.567	0.056
Currency	7	3	0.162	2.716	0.201
Email	5	1	0.044	0.714	0.087
Financial	3	1	0.067	1.089	0.095
Game	4	1	0.002	0.031	0.003
Geolocation	36	10	0.063	0.970	0.135
Languages	4	1	0.063	1.063	0.108
Login and security	15	5	0.024	0.394	0.042
Lookup	12	4	0.065	1.017	0.136
Math	15	5	0.107	1.850	0.200
News	18	5	0.034	0.539	0.081
Phone and SMS	17	6	0.057	0.901	0.093
Religion	3	1	0.012	0.190	0.017
Trade	10	3	0.044	0.677	0.096
Versioning	4	1	0.015	0.244	0.026
Weather	6	1	0.034	0.538	0.077
MEAN			**0.051**	**0.824**	**0.088**

Table 2: MEA, MAPE and RMSE results using the Neighborhood-based estimation

Category	MAE	MAPE(%)	RMSE	R	R2
News	0.4427	7.0962	0.4677	0.9211	0.8484
Currency	0.0755	0.7257	0.0057	0.9820	0.8633
Trade	0.8646	13.6597	0.9120	1.0000	1.0000
Math	0.1815	2.8735	0.2090	0.9364	0.8768
Login and security	0.7195	11.5262	1.1858	0.9152	0.8377
Geolocation	0.0751	1.2063	0.0991	0.7845	0.5741

Table 3: MEA, MAPE, RMSE, R and R^2 results using linear multiple regression

Services" shows the total number of services in the group, and column "Nbr tests" depicts the number of services considered as newcomers.

After several runs, we found that the program, with this working dataset, always finds Top-K elements for each service, with a small distance between reputation values of these Top-K elements. Thus, it always uses the neighborhood-based estimation technique (1). The obtained MEA, MAPE and RMSE are given in Table 2. We note that these values are the mean values of 10 execution rounds. We can see that the estimated reputations are very close to the actual values with MEA = 0.51 in average. In addition, Technique 1 estimates accurately the initial reputation with a percentage error of 0.824%, which is a very good score.

Besides, we applied the "Multiple Linear Regression"-based estimation technique (2) on some groups to test their estimation accuracy. To construct the multiple linear model we need a dataset of at least 9 services. Thus, we limited the use of this technique on groups with a high number of services. Test services were selected randomly, and the construction of the multiple linear model was with 80% of services.

Table 3 provides the obtained MAE, MAPE, RMSE, R and R^2. As we can see, there is a deviation between the estimated reputation values and the calculated values represented by MAE value (0.8464 for example in group "Trade"). This deviation is caused by the size of the trained data. All these groups hold basically a small number of services. However, these values are still close to the estimated values, and the percentage error does not exceed 14% in the worst case. The correlation coefficient (Multiple R) for all groups ranges in [0.78 - 1] which indicates a positive relationship between reputation values and QoS data (where 1 indicates a perfect positive relationship). We see also from R^2 that most of the trained values fit the model (e.g., for Math group 0.87% of the values fit the model). From this experiment, we can say that the largest is the data we use for training, the best accuracy could be achieved during the estimation of newcomers' reputation. We conclude that we may safely use this technique too to estimate the reputation of newcomers from their functionally-similar services.

7.7. SVR-based reputation

To construct the Support Vector Regression model, we used the LibSVM API [77] in our Java program. Since we want to construct a model that covers both linear and non-linear relationships between QoS and reputation values in the system, we choose to use the Gaussian Radial Basis function

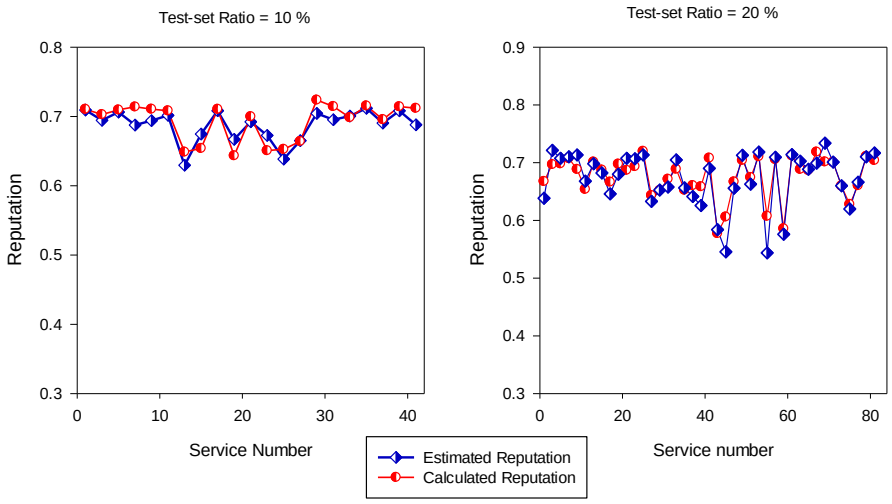

Figure 3: Comparison samples between calculated and estimated reputation with different test-set rates

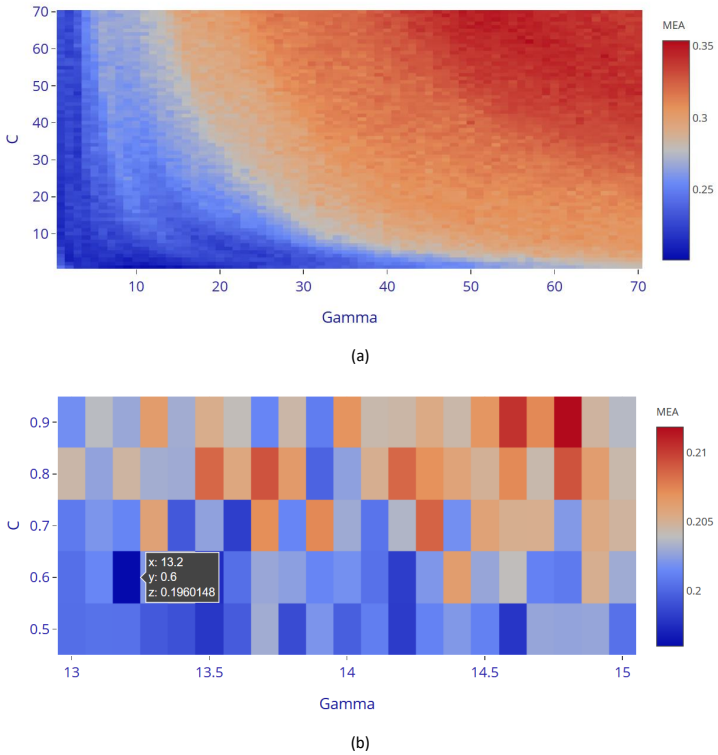

Figure 4: Effects of γ (gamma) and C on the performance of the SVR Model

(RBF). RBF is chosen because it is working well in practice, very easy to tune, and it is recommended by the machine learning community to use it as a default SVR kernel. In addition, linear kernels and polynomial kernels are a special case of Gaussian RBF kernel.

7.8. Tuning

In this section, we focus on selecting the best parameters for $\epsilon-$SVR (i.e. the best parameters that output the lower MEA). In fact, $\epsilon-$SVR with RBF Kernel has one hyper-parameter (σ) and two open parameters C (Cost or regularization parameter) and ϵ (epsilon or the insensitive coefficient), where:

- σ corresponds to the width or the scale of this Kernel.
- C: Cost $[0 \rightarrow \infty]$ represents the penalty associated with errors larger than epsilon. Increasing cost value causes closer fitting to the calibration/training data.
- ϵ: epsilon represents the minimal required precision.

LIBSVM allows the tuning of σ through γ, where $\gamma = 1/(2 * \sigma^2)$ which means a bigger gamma means smaller sigma in the formula and therefore smaller influence of a data point such that only those support vectors very close to the decision boundary are considered and therefore a bigger chance for overfitting the training data.

We have variated the γ parameter between 0.001 to 1000, and C between 1 and 1000, and ϵ is set to 0.0001 which is the minimal precision found after variating Epsilon from (0.0001 to 1). Figure 4 shows a heatmap that presents the performance of the model using different γ and C parameters. Only a part from the data is plotted in the maps (a and b). From this procedure, we found that the models performs well when γ ranges in the interval $[13 \rightarrow 15]$ and C ranges in the interval $[0.5 \rightarrow 08]$. Finally, $\gamma = 13.2$, $C= 0.6$, and $\epsilon = 0.0001$, are selected as the best parameter for building the SVR model.

7.9. Evaluation

To avoid losing important patterns in the dataset, which in turn increases error induced by bias, we use in this experiment a 10-fold cross validation; that is, the dataset is divided into 10 subsets, the evaluation is repeated 10 times such that in each time, one of the 10 subsets is used as the test set and the 9 remaining subsets are put together to train the SVR model. The MEA

Maliciousness Density (%)	Method					
	MDBM	NM	MVM	ARM	REM	ANN
10	1.0132	2.0503	1.0555	0.2061	**0.1066**	0.2211
20	0.2462	1.8053	1.0030	0.1504	**0.1378**	0.1785
30	0.5557	1.5522	0.6482	0.2436	**0.1482**	0.2167
40	1.7576	2.2996	1.4167	0.3702	**0.1582**	0.2512
50	1.3253	1.3160	0.6803	0.3815	**0.1876**	0.2307
60	2.8844	0.8443	1.7422	1.5413	**0.2532**	0.6324
70	3.8457	1.0757	2.0571	1.6836	**0.6539**	0.7673

Table 4: MEA comparison between different methods using various malicious and test-set densities

estimation is averaged over the 10 trials. Many runs have been performed varying the maliciousness rate performing for each variation a 10-fold cross validation as described above. A comparison between results obtained by our SVR-model and other methods are summarized in Table 4.

Before analyzing results in this table, we present in Figure 3, a sample of newcomers estimated reputation values versus the calculated reputation values for two test-sets (10% and 20% without cross-validation). As we can see, there is a small variation (less than 0.2 in the worst cases) between the SVR-estimated value and the actual value calculated by feedback ratings for a different number of services.

Besides, for the comparison results obtained by variating maliciousness density with 10-fold cross validation (depicted in Table 4), we can observe that our method (REM) gave the smaller MAE values (indicating a better accuracy) consistently. The ANN method gives results relatively close to REM. The ARM method gives also a good result with a MEA less than 0.25 when malicious density is less than 30In addition, we observe that when the malicious density exceeds 50% the accuracy of all methods decreases due to the high number of malicious users.

From these results, we may safely conclude that the proposed estimation method is more effective in assigning newcomers' reputation.

8. Conclusion

In this work, we proposed a method that estimates reputation values of newcomer web services. Initial reputation values assigned to newcomer

web services have an impact on the performance of web service recommendation systems. Our proposed method uses (i) provider reputation values, (ii) neighbor services' reputation values, and (iii) SVR-based reputation estimation model, trained by QoS and reputation values of long-standing web services, to correctly estimate reputation values of newcomer web services. This method addresses both the cold start and the whitewashing problems often encountered in online recommendation systems.

We conducted several experiments on real Web services to evaluate the efficiency of the proposed method. Through experimental evidence, we showed that the proposed method outperforms existing methods and may be safely applied to estimate the reputation of newcomers in Web service recommendation systems.

Although the proposed method and the conducted experiments considers only Web services, the proposed work can be used with any kind of service or API in general, Cloud, mobile, or Web (REST, WebSocket, SOAP, ...) ones, provided that we select a similarity assessment algorithm between service or API interface descriptions. This algorithm is responsible for comparing services and collecting the set of functionally-similar ones. The rest of the method is not impacted.

Our future research work includes the proposition of a complete service recommendation system with a unified reputation management and security model for both single, composite, and community-based Web services. These models target more online attacks such as the request drop, denial of service, outage, and eavesdropping attacks.

[1] Y. Wang, C. Guo, T. Li, Q. Xu, Secure Two-Party Computation in Social Cloud Based on Reputation, in: Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on, IEEE, 242–245, 2015.

[2] Z. Maamar, G. Costantino, M. Petrocchi, F. Martinelli, Business Reputation of Social Networks of Web Services, Procedia Computer Science 56 (2015) 18 – 25, ISSN 1877-0509, the 10th International Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) Affiliated Workshops.

[3] F. Moyano, C. Fernandez-Gago, J. Lopez, A Model-driven Approach

for Engineering Trust and Reputation into Software Services, Journal of Network and Computer Applications (2016) –ISSN 1084-8045.

[4] M. Mehdi, N. Bouguila, J. Bentahar, Trust and Reputation of Web services Through QoS Correlation Lens, IEEE Transactions on Services Computing PP (99) (2015) 1–1.

[5] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, F. Yang, Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems, IEEE Transactions on Services Computing 8 (5) (2015) 755–767, ISSN 1939-1374.

[6] T. H. Noor, Q. Z. Sheng, S. Zeadally, J. Yu, Trust management of services in cloud environments: Obstacles and solutions, ACM Comp. Surveys 46 (1) (2013) 12.

[7] F. Hendrikx, K. Bubendorfer, R. Chard, Reputation systems: A survey and taxonomy, Journal of Parallel and Distributed Computing 75 (2015) 184–197.

[8] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, A survey on trust and reputation models for Web services: Single, composite, and communities, Decision Support Systems 74 (2015) 121–134.

[9] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The adaptive web, Springer, 291–324, 2007.

[10] K. Huang, Y. Liu, S. Nepal, Y. Fan, S. Chen, W. Tan, A Novel Equitable Trustworthy Mechanism for Service Recommendation in the Evolving Service Ecosystem, in: Service-Oriented Computing, Springer, 510–517, 2014.

[11] Z. Malik, A. Bouguettaya, Reputation bootstrapping for trust establishment among web services, Internet Computing, IEEE 13 (1) (2009) 40–47.

[12] F. G. Mármol, M. Q. Kuhnen, Reputation-based Web service orchestration in cloud computing: A survey, Concurrency and Computation: Practice and Experience 27 (9) (2015) 2390–2412.

[13] J. R. Douceur, The sybil attack, in: Peer-to-peer Systems, Springer, 251–260, 2002.

[14] Z. M. Aljazzaf, Trust-based service selection, Ph.D. thesis, The University of Western Ontario, 2011.

[15] Q. Wu, Q. Zhu, P. Li, A neural network based reputation bootstrapping approach for service selection, Enterprise Information Systems 9 (7) (2015) 768–784.

[16] M. Chen, L. He, X. Cai, W. Xia, Trust evaluation model for composite service based on subjective logic, in: Proc. of IIHMSP'08, IEEE, 1482–1485, 2008.

[17] M. Feldman, J. Chuang, The evolution of cooperation under cheap pseudonyms, in: Proc. of CEC'05, IEEE, 284–291, 2005.

[18] O. Tibermacine, C. Tibermacine, F. Cherif, Regression-Based Bootstrapping of Web Service Reputation Measurement, in: Web Services (ICWS), 2015 IEEE International Conference on, IEEE, 377–384, 2015.

[19] Z. Malik, A. Bouguettaya, Rateweb: Reputation assessment for trust establishment among web services, VLDB Journal 18 (4) (2009) 885–911.

[20] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and computing 14 (3) (2004) 199–222.

[21] M. Awad, R. Khanna, Support Vector Regression, in: Efficient Learning Machines, Springer, 67–80, 2015.

[22] B. Tian, J. Han, K. Liu, Closed-Loop Feedback Computation Model of Dynamical Reputation Based on the Local Trust Evaluation in Business-to-Consumer E-Commerce, Information 7 (1) (2016) 4.

[23] D. Isherwood, M. Coetzee, Trust CV: Reputation-based trust for collectivist digital business ecosystems, in: Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, IEEE, 420–424, 2014.

[24] M.-H. Peetz, M. de Rijke, R. Kaptein, Estimating Reputation Polarity on Microblog Posts, Information Processing & Management 52 (2) (2016) 193 – 216, ISSN 0306-4573.

[25] J. Hu, Y. Zhang, Research patterns and trends of Recommendation System in China using co-word analysis, Information Processing & Management 51 (4) (2015) 329 – 339, ISSN 0306-4573.

[26] A. J. Bidgoly, B. T. Ladani, Benchmarking reputation systems: A quantitative verification approach, Computers in Human Behavior 57 (2016) 274–291.

[27] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, G. M. Sarné, A Distributed Reputation-Based Framework to Support Communication Resources Sharing, in: Intelligent Distributed Computing IX, Springer, 211–221, 2016.

[28] L. Barakat, S. Mahmoud, P. Taylor, N. Griffiths, S. Miles, Reputation-based provider incentivisation for provenance provision, in: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), 2016.

[29] W. Itani, C. Ghali, A. Kayssi, A. Chehab, Reputation as a Service: A System for Ranking Service Providers in Cloud Systems, in: Security, Privacy and Trust in Cloud Systems, Springer, 375–406, 2014.

[30] E. Majd, V. Balakrishnan, A trust model for recommender agent systems, Soft Computing (2016) 1–17.

[31] H. Zhao, X. Li, VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks, The Journal of Supercomputing 64 (3) (2013) 805–829.

[32] A. Louati, J. El Haddad, S. Pinson, A Multi-Agent Approach for Trust-based Service Discovery and Selection in Social Networks, Scalable Computing: Practice and Experience 16 (4) (2016) 381–402.

[33] B. Zhang, Q. Song, T. Yang, Z. Zheng, H. Zhang, A Fuzzy Collusive Attack Detection Mechanism for Reputation Aggregation in Mobile Social Networks: A Trust Relationship Based Perspective, Mobile Information Systems 2016.

[34] J.-H. Cho, A. Swami, R. Chen, A survey on trust management for mobile ad hoc networks, Communications Surveys & Tutorials, IEEE 13 (4) (2011) 562–583.

[35] S. Sutariya, P. Modi, A Review of Different Reputation Schemes to Thwart the Misbehaving Nodes in Mobile Ad Hoc Network., International Journal of Computer Science & Information Technologies 5 (3).

[36] W. Zheng, L. Jin, A Consumer Decision-Making Model in M-Commerce: The Role of Reputation Systems in Mobile App Purchases, Information Resources Management Journal (IRMJ) 29 (2) (2016) 37–58.

[37] Z. Yan, P. Zhang, A. V. Vasilakos, A security and trust framework for virtualized networks and software-defined networking, Security and communication networks 9 (16) (2016) 3059–3069.

[38] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, M. F. Zhani, Research Challenges in Nextgen Service Orchestration, arXiv preprint arXiv:1806.00764 .

[39] M. Kablan, C. Joe-Won, S. Ha, H. Jamjoom, E. Keller, The cloud needs a reputation system, arXiv preprint arXiv:1509.09057 .

[40] G. Sun, Y. Li, Y. Li, D. Liao, V. Chang, Low-latency orchestration for workflow-oriented service function chain in edge computing, Future Generation Computer Systems 85 (2018) 116–128.

[41] S. Betge-Brezetz, G.-B. Kamga, M. Tazi, Trust support for SDN controllers and virtualized network applications, in: Network Softwarization (NetSoft), 2015 1st IEEE Conference on, IEEE, 1–5, 2015.

[42] J. B. Bernabe, J. L. H. Ramos, A. F. S. Gomez, TACIoT: multidimensional trust-aware access control system for the Internet of Things, Soft Computing 20 (5) (2016) 1763–1779.

[43] S. Ravidas, S. Lal, I. Oliver, L. Hippelainen, Incorporating trust in NFV: Addressing the challenges, in: Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on, IEEE, 87–91, 2017.

[44] F. Rocha, M. Correia, Lucy in the sky without diamonds: Stealing confidential data in the cloud, in: Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on, IEEE, 129–134, 2011.

[45] B. K. Mughal, S. Hameed, G. M. Shaikh, A Centralized Reputation Management Scheme for Isolating Malicious Controller (s) in Distributed Software-Defined Networks, arXiv preprint arXiv:1711.11005 .

[46] K. Giotis, M. Apostolaki, V. Maglaris, A reputation-based collaborative schema for the mitigation of distributed attacks in sdn domains, in: Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP, IEEE, 495–501, 2016.

[47] Z. Liu, J. Ma, Z. Jiang, Y. Miao, C. Gao, IRLT: Integrating Reputation and Local Trust for Trustworthy Service Recommendation in Service-Oriented Social Networks, PloS one 11 (3) (2016) e0151438.

[48] J. Yao, W. Tan, S. Nepal, S. Chen, J. Zhang, D. D. Roure, C. Goble, ReputationNet: Reputation-Based Service Recommendation for e-Science, IEEE Transactions on Services Computing 8 (3) (2015) 439–452, ISSN 1939-1374.

[49] Z. Yan, X. Li, R. Kantola, Controlling Cloud Data Access Based on Reputation, Mobile Networks and Applications 20 (6) (2015) 828–839.

[50] C. Zhu, H. Nicanfar, V. Leung, L. T. Yang, An authenticated trust and reputation calculation and management system for cloud and sensor networks integration, Information Forensics and Security, IEEE Transactions on 10 (1) (2015) 118–131.

[51] G. Zacharia, A. Moukas, P. Maes, Collaborative reputation mechanisms for electronic marketplaces, Decision Support Systems 29 (4) (2000) 371–388.

[52] T. D. Huynh, N. R. Jennings, N. R. Shadbolt, Certified reputation: how an agent can trust a stranger, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, 1217–1224, 2006.

[53] X. Wang, K. Govindan, P. Mohapatra, Provenance-based information trustworthiness evaluation in multi-hop networks, in: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, IEEE, 1–5, 2010.

[54] S. Jin-dian, G. He-qing, G. Yin, An adaptive trust model of Web services, Wuhan University Journal of Natural Sciences 10 (1) (2005) 21–25.

[55] M. H. Hasan, J. Jaafar, M. F. Hassan, Monitoring web services quality of service: a literature review, Artificial Intelligence Review 42 (4) (2014) 835–850.

[56] O. Tibermacine, C. Tibermacine, F. Cherif, A Practical Approach to the Measurement of Similarity between WSDL-based Web Services, Revue des Nouvelles Technologies de l'Information 6th French-speaking Conference on Software Architectures, RNTI-L-7 (2014) 03–18.

[57] X. Zheng, L. Da Xu, S. Chai, QoS Recommendation in Cloud Services, IEEE Access 5 (2017) 5171–5177.

[58] M. Garriga, A. Flores, C. Mateos, A. Zunino, A. Cechich, Service selection based on a practical interface assessment scheme, International Journal of Web and Grid Services 9 (4) (2013) 369–393, ISSN 1741-1106.

[59] N. Kokash, A comparison of web service interface similarity measures, Frontiers in Artificial Intelligence and Applications 142 (2006) 220, ISSN 0922-6389.

[60] O. Tibermacine, C. Tibermacine, F. Cherif, WSSim: a Tool for the Measurement of Web Service Interface Similarity, in: French-speaking Conference on Software Architectures (CAL'13), 2013.

[61] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender Systems: An Introduction–Cambridge University Press, New York, 2010.–352 P .

[62] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 143–177.

[63] Z. Zheng, H. Ma, M. R. Lyu, I. King, Collaborative web service qos prediction via neighborhood integrated matrix factorization, Services Computing, IEEE Transactions on 6 (3) (2013) 289–299.

[64] H. Ma, I. King, M. R. Lyu, Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th annual international ACM

SIGIR conference on Research and development in information retrieval, ACM, 39–46, 2007.

[65] R. Chen, C.-Y. Liang, W.-C. Hong, D.-X. Gu, Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm, Applied Soft Computing 26 (2015) 435–443.

[66] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, A. Fouilloy, Machine learning methods for solar radiation forecasting: A review, Renewable Energy 105 (2017) 569–582.

[67] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, N. Zerhouni, Direct remaining useful life estimation based on support vector regression, IEEE Transactions on Industrial Electronics 64 (3) (2017) 2276–2285.

[68] W. Zhao, T. Tao, E. Zio, System reliability prediction by support vector regression with analytic selection and genetic algorithm parameters selection, Applied Soft Computing 30 (2015) 792–802.

[69] J. Hu, J. Qi, Y. Peng, Q. Ren, Predicting electrical evoked potential in optic nerve visual prostheses by using support vector regression and case-based prediction, Information Sciences 290 (2015) 7–21.

[70] A. Kavousi-Fard, H. Samet, F. Marzbani, A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting, Expert systems with applications 41 (13) (2014) 6047–6056.

[71] Z. Zheng, Y. Zhang, M. R. Lyu, Distributed qos evaluation for real-world web services, in: Proc. of ICWS'10, IEEE, 83–90, 2010.

[72] E. Al-Masri, Q. H. Mahmoud, Qos-based discovery and ranking of web services, in: Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on, IEEE, 529–534, 2007.

[73] H. T. Nguyen, J. Yang, W. Zhao, Bootstrapping trust and reputation for Web services, in: Commerce and Enterprise Computing (CEC), 2012 IEEE 14th International Conference on, IEEE, 41–48, 2012.

[74] N. Limam, R. Boutaba, Assessing software service quality and trustworthiness at selection time, IEEE Transactions on Software Engineering, 36 (4) (2010) 559–574.

[75] R. L. Oliver, A cognitive model of the antecedents and consequences of satisfaction decisions, Journal of marketing research (1980) 460–469.

[76] A. Whitby, A. Jøsang, J. Indulska, Filtering out unfair ratings in bayesian reputation systems, in: Proc. 7th Int. Workshop on Trust in Agent Societies, vol. 6, 2004.

[77] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.

Author Biographies

Okba Tibermacine is an assistant professor in computer science at Biskra University, Algeria. He graduated from Biskra university with a B.Eng. degree. He received his M.Sc. degree in computer science from Batna University, Algeria and Ph.D. degree from Biskra University, Algeria in 2015. His current research interests include Service selection and recommendation, reliability analysis, error-handling and self-healing Web service compositions.

Chouki Tibermacine is an associate professor at Montpellier University (France) since fall 2007. He received his Ph.D. from the University of South Brittany (France) in 2006 and M.Sc in Distributed Systems from the University of Paris VI (France) in 2003. His current research focuses on the specification, evolution and transformation of object-oriented, component-based and service-oriented software architectures and programs. He participated to several research projects with industrial (IBM among others) and international academic partners. He co-authored about thirty peer-reviewed articles. He was the publicity chair of ECOOP'13, ECSA'13 and ECMFA'13 organized jointly in Montpellier in 2013. Since 2010, he is co-responsible for the French work group on software reverse-engineering, maintenance and evolution (GTRIMEL) of the french CNRS research group on Programming and Software Engineering (GDR GPL). He received the ACM SIGSOFT Distinguished Paper Awards in CBSE'11 and CBSE'14. He is holding the Scientic Excellence Fellowship from Montpellier University for the period 2012-2016.

Foudil Cherif is an Associate Professor of computer science at Computer Science Department, Biskra University, Algeria. Dr. Cherif holds PhD degree in computer Science. The topic of his doctoral dissertation is Behavioral Animation: simulation of a crowd of virtual humans. He also possesses B. Sc. (engineer) in computer science from Constantine University 1985, and an M.Sc. in computer science from Bristol University, UK 1989. He is currently the head of LESIA Laboratory. His current research interest is in artificial intelligence, artificial life, crowd simulation and software engineering.