Computer Science and Information Systems 19(2):709–733 https://doi.org/10.2298/CSIS210518054M

QoS Prediction for Service Selection and Recommendation with a Deep Latent Features Autoencoder

Fatima Zohra Merabet and Djamel Benmerzoug

LIRE Labotory, Faculty of NTIC,
University of Constantine2-Abdelhamid Mehrin Constantine,
Algeria
{$fatima.merabet, djamel.benmerzoug$}@univ-constantine2.dz

Abstract. The number of services on the Internet has increased rapidly in recent years. This makes it increasingly difficult for users to find the right services from a large number of the functionally equivalent candidate. In many cases, the number of services invoked by a user is quite limited, resulting in a large number of missing QoS values and sparseness of data. Consequently, predicting QoS values of the services is important for users to find the exact service among many functionally similar services. However, improving the accuracy of QoS prediction is still a problem. Despite the successful results of the proposed QoS prediction methods, there are still a set of issues that should be addressed, such as Sparsity and Overfitting. To address these issues and improve prediction accuracy. In this paper, we propose a novel framework for predicting QoS values and reduce prediction error. This framework named auto-encoder for neighbor features (*Auto-NF*) consists of three steps. In the first step, we propose an extended similarity computation method based on Euclidean distance to compute the similarity between users and find similar neighbors. In the second step, we form clusters of similar neighbors and partition the initial matrix into sub-matrices based on these clusters to reduce the data sparsity problem. In the third step, we propose a simple neural network autoencoder that can learn deep features and select an ideal number of latent factors to reduce the overfitting phenomenon. To validate and evaluate our method, we conduct a series of experiments use a real QoS dataset with different data densities. The experimental results demonstrate that our method achieves higher prediction accuracy compared to existing methods.

Keywords: similarity computation, neighbors selection, quality of service (QoS), QoS prediction, autoencoder.

1. Introduction

Service recommendation and selection have attracted much attention in the service computing community in recent years [4]. With the dramatic increase in the number of services, different service providers offer many services with the same or similar functions [45]. At the same time, due to the large number of available services, it becomes more difficult for a user to select services that meet his or her requirements [43]. So, it is an urgent task to solve how to recommend suitable services to meet users' requirements. The key criterion considered in recommending services is their Quality of Service (*QoS*), which

can distinguish the suitable services among different functionally equivalent services [45]. Most previous studies have assumed that the quality values of candidate services must be known and accurate. However, it is really hard for a user to invoke all candidate services to acquire their QoS values and make a final decision [45]. Thus, QoS prediction is an indispensable task to finish service selection and recommendation with high quality.

As mentioned above, an active user usually can use only a limited number of services due to the enormous number of them on the Internet what makes the QoS data very sparse (*there are many entries without QoS values*). As a result, the task of QoS prediction to complete the unknown entries in the dataset is really important. Traditional Collaborative Filtering based methods are used to predict missing values. These last based on: 1) Similarity Calculation. Generally, the similarity is calculated using the known QoS values between users and services. And, the widely adopted similarity computational model includes Pearson correlation coefficient, cosine. etc [27]. 2) Neighborhood Selection. In this step, similar neighbors of users and services are identified based on the computed similarities [27]. 3) Collaborative Prediction. Here, the final prediction of QoS values is made by weighting the sum of QoS values of the selected neighbors [27]. Among all prediction methods, collaborative filtering (*CF*) methods have been deeply studied and applied mainly because of their simplicity and effectiveness. However, In QoS prediction, high data sparsity is a common problem and neighborhood-based CF method is not able to learn latent features from historical QoS records. Therefore, the community has proposed a new model-based CF algorithm that attracts more attention to latent features learning. As a typical latent factor model, matrix factorization (*MF*) achieves good performance in learning latent features in high sparsity. Many existing studies extend MF for QoS prediction. However, despite the successful results of MF in the recommendation area, there are still a set of problems that should be handled, as we will mention in the sect. 2. In this study, we aim to address two major issues: 1) the sparsity caused by the service invocation matrix. 2) The overfitting due to the latent factors learning. These two main problems affect prediction accuracy. Therefore, improving the accuracy of QoS prediction has become a challenge.

Recently, deep neural networks have received much attention in many fields and have become increasingly popular [45]. However, few studies use neural network techniques in QoS prediction. One of the most widely used deep learning methods is autoencoder due to the advantages of fast convergence and no labeling requirement. Autoencoder has a simple network structure and also has a strong ability to learn latent features [45].

In this paper, we propose a framework-based autoencoder to alleviate the previous issues and exploit the benefits of improving the quality of neighborhood selection and learning deep latent features from the QoS dataset. The purpose of our model is to minimize the discrepancy between input and output data. The main contributions of our work can be summarized as follows:

- we propose an extended similarity computation method based on Euclidean distance to compute the similarity between users and find similar neighbors. In this method, we use a simple and efficient concept based on common services invoked by both users to improve and facilitate the quality of neighbors' selection;
- we create clusters for similar neighbors according to the computed similarities based on QoS values. And, we partition the initial matrix into small sub-matrices based on these clusters to reduce the data sparsity problem;

- we propose a neural network autoencoder capable of learning deep features and selecting an ideal number of hidden neurons to reduce the overfitting phenomenon in the learning step;
- to validate and evaluate our method, we conduct a series of experiments use a real QoS dataset with different data densities. The experimental results show that our method achieves higher prediction accuracy compared to existing methods.

The remaining sections of this paper are organized as follows. Sect. 2 summarizes related work on QoS prediction. Sect. 3 explains the whole framework. Sect. 4 elaborates an example of motivation. Sect. 5 gives the experimental results. Sect. 6 discusses our results, and sect. 7 concludes the paper.

2. Related Work

To improve the prediction accuracy, many researchers have proposed a series of QoS prediction methods. These can be broadly classified into two categories, i.e., collaborative filtering (*CF*) methods and content-based methods. Most of the proposed methods are extended from CF algorithm [3], [34]. In this section, we introduce a literature review about CF-based QoS prediction method, deep learning-based QoS prediction method, and autoencoder based QoS prediction method.

2.1. CF based QoS Prediction

Collaborative filtering (*CF*) is widely used in web service recommendation and service selection [16] due to its good performance. The basic idea of collaborative filtering is to use historical data for prediction [20][23]. Generally, CF-based QoS prediction methods can be divided into two categories, including neighborhood-based CF and model-based CF [2]. The neighborhood uses the existing QoS values of similar users (*or services*) to predict the missing QoS values, whereas model-based CF approaches build a predefined prediction model trained using historical QoS data to then predict missing QoS values. The improvement of most neighborhood-based CF methods is done by improving or designing new similarity calculation techniques, improving the quality of neighbor selection or combining different methods.

The neighborhood-based CF methods are widely used in QoS prediction and have the advantages of easy implementation and high scalability. Neighborhood-based CF algorithms can be classified into user-based CF methods, item-based CF methods and hybrid CF methods. Nilashi et al. [17] propose a new hybrid recommendation method based on Collaborative Filtering (CF) approaches using dimensionality reduction (SVD) and ontology techniques. Shao et al. [26] proposed a user-based CF with a new user similarity calculation method. Sun et al. [30] proposed a new similarity method for calculating the similarity between two services. Tang et al. [31] proposed a hybrid method to find similar neighbors in users set and services set respectively. Zheng et al. [30] introduced an improved similarity computation method to find similar neighbors.

This category mainly suffers from the data sparsity problem due to the limited number of Web services that a single user invokes, suffers from cold start problems [49][26] i.e., how to recommend a service to a user when there are few or no QoS records.

To alleviate the limitations of neighborhood methods, the community has designed model-based methods that can deal with the data sparsity problem. Papadakis et al. [18] propose a Synthetic Coordinate based Recommendation system. It is parameter-free, so it does not require tuning to achieve high performance and is more resistant to the cold start problem compared to other algorithms. Model-based CF methods use machine learning techniques, such as latent factor models [45], clustering-based models [41], [44], and aspect-based models [28]. Different form neighborhood-based methods that make prediction directly from the ratings of similar neighbors, model-based methods compress user-service rating matrix into a low-dimensional representation in terms of latent features using matrix factorization (*MF*) technique. The most widely applied algorithm that factorizes the rating matrix into two matrices: the user‘s feature matrix and the item's feature matrix, where one row and one column of each matrix are taken as the inner product for prediction [9]. Matrix factorization has emerged as one of the main approaches of model-based method because it can handle sparse matrices and produces good prediction accuracy. It has been verified by many experiments that model-based methods often have better performance than neighborhood-based methods. However, the existing model-based methods can only learn linear relationships between a user and a service, and the inner product cannot catch deep features [6]. Xu et al. [40] introduced a probabilistic matrix factorization model into service recommendation. Chen et al. [5] proposed a fuzzy clustering-based approach to learn latent features of users and services and designed a latent factor model to learn the features of each cluster. Mattiev et al. [15] propose new methods capable of reducing the number of class association rules generated by "classical" classifiers for class association rules, that use distance-based agglomerative hierarchical clustering. Zhu et al. [51] proposed a new context-aware reliability prediction approach, which solves the problem of data sparsity by constructing context-aware reliability models. Wu et al. [36] proposed a deep latent factor model by sequentially connecting multiple latent factor models.

2.2. Deep Learning based QoS Prediction

Recently, deep neural networks have received much attention in many fields and have become increasingly popular. Compared to the pure CF or MF methods, the neural network's method achieve better performance in traditional recommender systems. Some studies have attempted to integrate neural networks into collaborative filtering [1], [10]. However, there are still few studies using neural network techniques in QoS prediction. Jin et al. [8] proposed a deep learning model for predicting QoS of Web service, which builds the model through multi-layer perceptron (*MLP*) and convolution neural networks (*CNN*). Paradarami et al. [19] also presented a deep learning framework to recommend new products to users.

2.3. Autoencoder based QoS prediction

As one of many deep learning methods, autoencoder model is widely applied in recommendation systems for the advantages of fast convergence and no labeling requirement. The auto-encoder has a simple network structure and a strong ability to learn latent features. The autoencoder [24], [7] is an unsupervised neural network and can encode itself

with its latent factors. For example, Liang and Baldwin [13] utilized an autoencoder model to learn the user latent feature matrix for achieving fairly good performance in recommendation. Zhuang et al. [52] proposed a Dual-Autoencoder model to generate latent user and item feature matrices. The existing autoencoder based recommendation systems can be generally divided into two types [46]: one is to learn the latent feature representations of users and items as [21], and the other is to fill the missing value of the original matrix in the reconstruction layer of autoencoder. The authors in [33] tried to use the denoising autoencoder to reduce features dimension. Yin et al. use the autoencoder improved by the substitution strategy to obtain nonlinear latent features of users and services, and missing QoS are generated by the traditional MF methods [45]. Since autoencoder generally uses one hidden-layer neural network to learn the embedding feature, Zhang et al. adopt the MLP to model the nonlinear characteristics of embedding features [47], they also embed similar neighborhoods in MLP to further improve prediction accuracy [8]. Wu et al. proposed a deep neural network for making QoS prediction with contextual information [38], where a deep neural network is added to the end of FM in series for prediction.

The ultimate goal of our work is to solve problems that are complementary to those addressed by the previous works, like data sparsity and overfitting and obtain the deep hidden features by autoencoders. The experimental results presented in sect. 5 demonstrate that Auto-NF significantly outperforms the existing compared methods.

3. The Proposed Framework

Some research works deal with QoS prediction to recommend high-quality services to users in a dynamic environment. However, these works ignore the effect of data sparsity and overfitting and cannot learn deep features. To overcome the major challenges, we propose a high-reliability QoS prediction framework based on an autoencoder neuronal network as shown in (Fig. 1). This framework named autoencoder of neighbor features for short Auto-NF is capable of learning the hidden features to achieve highly accurate QoS prediction.

As shown in (Fig. 1), we assume that the user-service QoS record is the official input of the proposed method. First, we convert this dataset into a matrix with n services in columns and m users in rows. Then, we calculate the similarity between users to obtain neighbors and create sub-matrices with these neighbors. After that, we add features to the matrices of neighbors. Finally, we predict missing QoS values of the services.

3.1. The Framework Procedure

The framework procedure consists of three stages. Stage 1 selects the neighbors of users and services. Stage 2 adds features to the sub-matrices of neighbors. Stage 3 predicts the missing QoS values. The three stages are explained in detail below:

Phase 1 (*neighbors selection*). This step is to identify the similar neighbors of users and services to partitions them into clusters.

1. Similarity calculation: we propose an improved similarity computation method based on common services invoked by both users. This similarity uses historical QoS

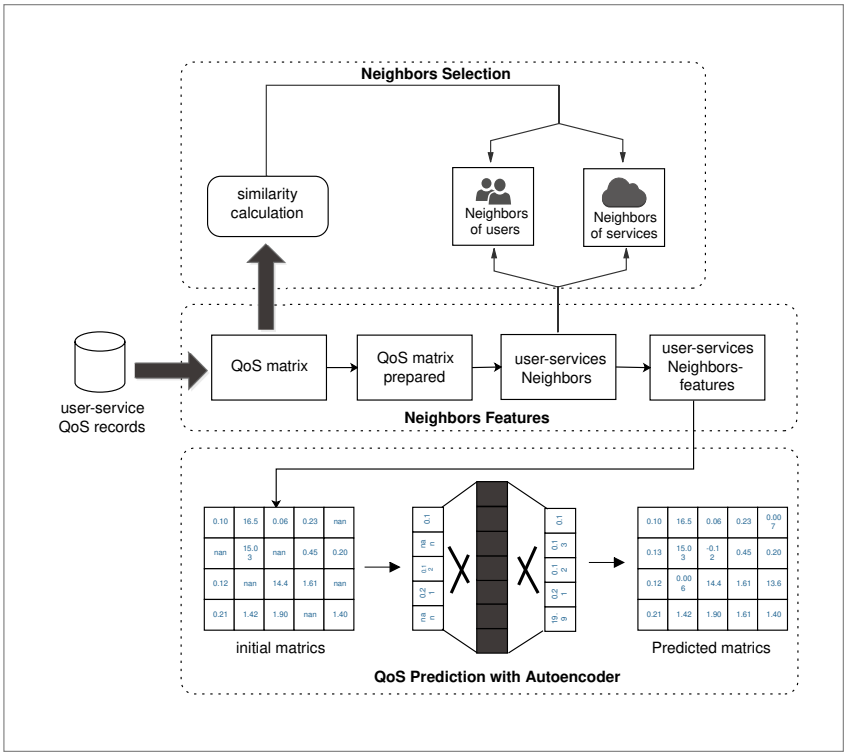

Fig. 1. The proposed framework Auto-NF for QoS prediction

records of services invoked by the same users. And, it can be determined by the two following factors:

- **Common services with different QoS values,** i.e., how many services that users have both invoked in the past;
- **Common services with the same QoS values,** i.e., how similar their QoS values are for services that users have both invoked in the past.

Based on these two factors, we incorporate the following formulate $\lambda_{u,v}$ into the similarity computation method to improve it and then increase the accuracy of neighbor's selection. This formula is computed as:

$$\lambda_{u,v} = \log \frac{|i|}{\left|i_{u,v}\right|} \tag{1}$$

Where $|i|$ is the total number of services, and $\left|i_{u,v}\right|$ represents the number of common services invoked by any two users u and v.
We will now discuss how to consider the two factors defined.[1]

- **First factor.** When two users have both called a large number of services. They have partitioned into the same cluster. And they will likely invoke similar other services;
- **Second factor.** Given a service s invoked by user u and user v, s is considered as a common service for these two users if their properties for service s are similar. If the number of common services is large, they are classified into the same cluster.

The proposed similarity is based on Euclidean distance and computed as follows:

$$S_{u,v} = \frac{1}{1 + \sqrt{\frac{\sum_{i=0}^{M} ((q_{u,i} - \bar{q}_u) - (q_{v,i} - \bar{q}_v))^2 . \lambda_{u,v}}{|M|}}} \tag{2}$$

Where $S_{u,v}$ is the similarity between user u and user v. We form the common behavior service invocation of services co-invoked like: $M = M_u \cap M_v$ which is the set of services invoked by both user u and user v. M_u denotes the set of services invoked by user u and M_v denotes the set of services invoked by user v. $q_{u,i}$ is the real QoS value generated after the target service i is invoked by the target user u, and $q_{v,i}$ is the real QoS value of the target service i is invoked by the target user v. We add a one to the similarity formula to avoid division by zero. $\bar{q}_u$ is the average QoS value of user u, and $\bar{q}_v$ is the average QoS value of user v. The final similarity results are recorded in the clusters of user and service neighbors.

2. Clusters selection: the goal of this step is to partition similar users and services into clusters. To determine reliable clusters, we divide the similarity values into ten intervals of [0, 0.1], [0.1, 0.2], ..., [0.9, 1], and [0, 0]. The users with the same similarity interval are included in the same cluster, and the services invoked by their users are partitioned into

[1] Remark

- Services accessed by the same users are considered as neighbors and placed in the same cluster.
- The services not invoked by any users are not considered;
- The users who have invoked few services and have no shared services with other users are considered untrusted users and not included in any cluster.

the same cluster according to the similarity interval. Each user appears in only one cluster. In this way, we obtain six clusters for users and six clusters for services, i.e., {u1, u6, u8}, {u9}, {u2, u4}, {u3, u5, u7}, {u15,u14}, and {s1, s0, s6}, {s0, s9, s8}, {s1,s2, s4}, {s3, s5, s7},{s0,s2,s20, s13}. The users or services in the same cluster have similar or even the same experiences with certain services. The final clustering results are recorded in the sub-matrices of neighbors.

Phase 2 (*neighbors features*). In the initial matrix, there are n services in columns and m users in rows, where each cell represents a corresponding QoS value assigned to a service by a user. We assume that these QoS contain missing values, resulting in a very sparse matrix. This step aims to reduce this sparsity by preparing the matrix and splitting it into pre-completed sub-matrices of neighbors and features because the denser the matrix, the better the results. To do this, we go through a series of steps described in the sect. 5 of results.

1. Matrices of neighbors: we divide the prepared matrix into sub-matrices that have a different number of rows and columns. The number of rows and columns corresponds to the size of each cluster. thus, each cluster of neighbors is represented with a reduced matrix. Finally, we add features to these sub-matrices and obtain six sub-matrices of neighbors.

2. Matrices of features: simultaneously with the calculation of similarity between users, six features vectors with the same number of clusters are generated. Here each vector represents a particular cluster of neighbors. The size of the vector equals six. As shown in Table 1, a vector initialized with zero receives a one at the specified cell, based on the specified similarity interval for that cluster. After that, the vectors of features and the matrices of neighbors are combined to create the neighbor's features matrices. Then, transmit these matrices to predict their missing QoS values with our autoencoder model.

Table 1. Features of clusters (*users and services*)

Similarity interval	Vectors of features						Clusters of users	Clusters of services
0.2–0.5	1	0	0	0	0	0	Clu_0	Cls_0
0.5–0.6	0	1	0	0	0	0	Clu_1	Cls_1
0.6–0.7	0	0	1	0	0	0	Clu_2	Cls_2
0.7–0.8	0	0	0	1	0	0	Clu_3	Cls_3
0.8–0.9	0	0	0	0	1	0	Clu_4	Cls_4
$0.9-1$	0	0	0	0	0	1	Clu_5	Cls_5

Phase 3 (*QoS Prediction with an Autoencoder*). Autoencoder is an unsupervised model that attempted to reconstruct the input data in the output layer [46] for each training sample through the network. In general, autoencoder can be considered as an extended version of artificial neural network with three or more layers (*an input layer, one or more hidden layers, and an output layer*), where the output layer should have the same size as the input layer. In our work, we propose an improved auto-encoder (an auto-encoder with SBS structure, sect. 3.2) to predict missing QoS values where the numbers of neurons in the

input and output layer are represented by the number of users. The learned autoencoder produces the missing QoS values and shown us the prediction error for these values.

3.2. Structure of Autoencoder

Autoencoder has two types of Structures as shown in (Fig. 2): Small-Big-Small (*SBS*) structure, where the largest layer is in the middle and the smallest layers are at the beginning and end. And Big-Small-Big (*BSB*) structure, where the smallest layer is in the middle and the largest layers are at the beginning and end. We tried both structures on our work and found that using a single hidden layer with a large number of neurons is the best (*SBS*).

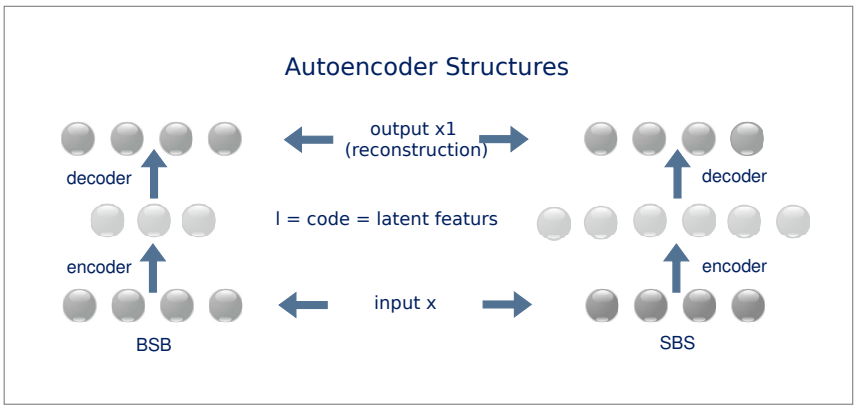

Fig. 2. General structures of autoencoder

4. Motivation and Problem Description

Our method aims to predict missing QoS values with high accuracy so that users can select the optimal service among candidate services. Table 2 shows an example of a matrix with 4 users in columns and 4 services in rows in total. In this matrix, each entry (*e.g.* $q_{1,1}$ *to* $q_{4,4}$) represents a property of the quality value (*e.g. In this paper, we mainly focus on the response time property*). We define the response time as the time duration between a user sending a request and receiving a response of a service (*e.g. Service i1 to Service i4*) observed by a user (*e.g. User u1 to User u4*) in the past. For example, when User u1 invokes Service i1, this response time value is recorded as the first entry $q_{1,1}$. The NaN value in the matrix means that the user has not invoked the service yet. And therefore, there is no record of the quality of service. Since users invoke few services in the real world, the user-service matrix is very sparse. The problem is how to use the known entries in predicting the unknown entries in the user-service matrix. More formally, the problem studied in this paper is defined as follows:

Given a sparse user-service matrix M, the existing entries $val = q_{u,i}$ are used to predict the missing (*NaN*) values. We define the users that invoked the same set of services as S(u) and the services that are commonly invoked by the same set of users as S(i).

Given a user u, a service i, a set of users U = {u1, u2, u3, u4}, and a set of services I = {i1, i2, i3, i4}, the prediction process of the quality of service i mainly consists of three parts: (1) identifying S(u) from U and S(i) from I; where S(u) is the user's neighbors and S(i) is the service's neighbors. (2) Creating the set matrices of features neighbors based on the initial matrix M and the set of neighbors S(u) and S(i). (3) Predict missing QoS values and know the optimal service preferred by user u from the candidate services in matrix M.

Table 2. An example of a user-service matrix

	Service i1	Service i2	Service i3	Service i4
User u1	$q_{1,1}$	$q_{1,2}$	NaN	$q_{1,4}$
User u2	NaN	$q_{2,2}$	$q_{2,3}$	$q_{2,4}$
User u3	$q_{3,1}$	$q_{3,2}$	NaN	$q_{3,4}$
User u4	NaN	NaN	$q_{4,3}$	NaN

From Table 2, we take User u1 as an example. The QoS value of Service i3 is missing for User u1. In this case, we represent it as $q_{1,3}$ and consider User u1 as the target user and Service i3 as the target service. So, u1 only invoke {i1, i2, i4} and we represent its QoS values for these services by $\{q_{1,1}, q_{1,2}, q_{1,4}\}$. While i1 invoked by {u1,u3}, i2 invoked by {u1, u2, u3}, and i4 invoked by {u1, u2, u3}. If we want to predict how user u1 will show the quality of service i3, we need to identify its similar users by calculating the similarities between him and three other users, namely u2, u3, and u4. First of all, we should find the sets of calls for these users. Then, based on these sets we can find the set of neighbors: From the table, we have seen that u2 invoke {i2, i3, i4}, u3 invoke {i1, i2, i4}, and u4 invoke {i3}. We represent their QoS values by $\{q_{2,2}, q_{2,3}, q_{2,4}\}$, $\{q_{3,1}, q_{3,2}, q_{3,4}\}$, and $\{q_{4,3}\}$ respectively. We have seen that u2 and u3 have several commons services with u1. Therefore, we can conclude that by $u1 \cap u3 = \{i1, i2, i4\}$ and $Su1 \cap u2 = \{i2, i4\}$, which means that both users u1 and u3 have used services i1, i2, i4. And both users u1 and u2 have used services i1 and i4, respectively. So, S(u) = u1,u2,u3 and S(i) = i1,i2,i4. In this way, we can obtain $q_{1,3}$, considering the similarity between u1 and u3 as: $\{q_{1,1}, q_{1,2}, q_{1,4}\}$, $\{q_{3,1}, q_{3,2}, q_{3,4}\}$, and between u1 and u2 as: $\{q_{1,2}, q_{1,4}\}$, $\{q_{2,2}, q_{2,4}\}$. In the same way, the remaining missing QoS values can be predicted. Considering u1 and u4, we obtain $S(u1) \cap S(u4) = \{\}$, which means that users u1 and u4 have any services in common. Based on the two factors defined in sect. 3, u1 and u4 are not similar. Therefore, u4 is not helpful and should not be used to predict $q_{1,3}$. Consequently, identifying similar users is of great significance. Meanwhile, it is also important to exclude dissimilar users.

5. Experimental and Evaluation

To evaluate the effectiveness and efficiency of our model, in this section we conducted a set of experiments on a real QoS dataset of Web services. Sect. 5.1 presents the experimen-

tal setup in detail, including parameters settings and dataset description. Sect. 5.2 shows the preparation of the dataset. Sect. 5.3 and sect. 5.4 introduce the methods comparison and the metrics used, respectively. Then, sect. 5.5 analyzes the results, including the evaluation of the table comparison, data validation, prediction, the effect of matrix density, and an example to evaluate our model with another type of dataset called MovieLens. In particular, our experiments aim to answer the following research questions (*RQs*): RQ1: How does the proposed method perform compared to the well-known state-of-the-art QoS prediction methods? Does it provide better prediction accuracy than them? RQ2: Does our model sensitive to over-fitting? RQ4: The data density is usually used to simulate the true scenario in the real world, where the goal is to evaluate the prediction accuracy of the proposed method. So, in our case, what is the performance of the proposed method under different data densities?

5.1. Experimental Setup

Our source code is implemented in python, runs on spyder, and all variants of experiments are performed on a machine with Intel(*R*) Core(*TM*) i7-8550U @ 1.80GHz CPU 1.99GHz with 8Go RAM on a Windows 10 Professional server.

Parameters Settings the parameters used in our approach are shown in Table 3.

Table 3. Important parameters used in our approach

Parameters	**Optimal value**
Regularization	0.001
Learning rate	0.0001
Activation function	Selu
Optimization function	Adam
input and output layers	The number of users in each matrix
Batch size (*hidden layer*)	2048
Layers	[input layer, hidden layer, output layer]

Dataset Description we conducted our experiments using one of the most widely available dataset in the world called WS-DREAM [2]: dataset#2, which was previously collected by Zheng et al [48]. WS-DREAM contains a total of 1,974,675 QoS records obtained from 4500 web services accessed by 142 users from all over the world. There are two attributes for each QoS record in this dataset, response time (*RT*) and throughput (*TP*). However, in this work, we focus on the response time dataset as the evaluation attribute. In total, we have $142 \times 4500 \times 64$ QoS records for each criterion (*response time or throughput*). This dataset has been used in the research community by many researchers [45],[43],[34] to evaluate the accuracy of QoS prediction. The statistics of our experimental dataset are shown in the following Table 4.

[2] https://wsdream.github.io

Table 4. Dataset statistics

Property Value	**RT**	**TP**
Range	0 20s	0 7000kbps
Average	1.33s	11.35kbps
Time slices function	64	64
Time interval	15min	15min
Number of users	142	142
Number of services	4500	4500
Number of all values	30 287 611	30 287 611
Number of missing values	10 609 313	10 609 313
Mean values	3.165s	9.608 kbps

5.2. Dataset Preparation

In real-world service invocation, the number of known QoS records is quite limited. Users typically invoke only a very small number of services, resulting in limited overlap of the same service invocation among users. However, in experiments, a user always invokes thousands of web services. Note that the original dataset is a dense full matrix. To simulate the real scenario and make the data-sparse, we randomly removed some records to obtain a density of 5% to 90%. Then, the sparse data is used as a training set, while the removed QoS value is used as a test set. For example, data with a density of 20% means that 20% of the data is kept as training set and used to build the predictive model, while the removed 80% of the data is used to evaluate the model. This step aims to reduce the number of invalid entries and thus improves the prediction accuracy. In the beginning, our dataset contains missing values. To precompute it, we first need to identify each missing $QoS_{i,u,s,t}$ value in the dataset and then replace it according to the following two rules:

$$QoS_{i,u,s,t} = average \tag{3}$$

$$QoS_{i,u,s,t} = 0 \tag{4}$$

Where $i\epsilon\{response\ time\}$, u is the user ID, s is the service ID, and t is the time slot of the QoS value ($t\epsilon\{0, ..., 63\}$) [29].

We make a comparison between zero and average values as shown in Table 5. We take the smallest and the largest matrix from our set of sub-matrices (matrix 0 and matrix 2) and calculate the error values predicted for both sub-matrices at the highest and lowest densities, and we take the average error values between the two sub-matrices. Where:
Matrix 0: is the smallest matrix in our set of sub-matrices with a size of 17 users in the rows and 4506 services in the cols.
Matrix 2: the largest matrix with a size of 38 users in the rows and 4506 services in the cols.

From Table 5 we can see that zero gives the smallest error of the predicted values. So, the results are better when we replace the missing values with zero than when we replace them with the average values.

In the following, we assume that this prepared dataset will be the official input of our proposed autoencoder model, and now we are ready to start learning it.

Table 5. Comparison between zero and average QoS value based on RMSE and MAE errors

		Training set density —— response time dataset							
		zero				average			
		10%		90%		10%		90%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Matrix 0	Prediction	0.1121	0.1644	0.0200	0.0287	0.1338	0.1808	0.0236	0.0328
Matrix 2	Prediction	0.1774	0.2416	0.0511	0.0711	0.2044	0.2708	0.0554	0.0761
average	Prediction	0.1447	0.2030	0.0356	0.0499	0.1691	0.2258	0.0395	0.0544

5.3. Comparison

To evaluate the prediction accuracy of our model, we compared it with several well-known QoS prediction models, as listed below.[3] The results are shown in Table 6 and Table 7.

UMEAN (*User Mean*): this method uses the average QoS value known by the active user to predict the missing QoS value of all services invoked by him [45].

IMEAN (*Item Mean*): this method employs the average QoS value on the used services to predict the QoS values of the unused services [45].

UPCC is a user-based Collaborative Filtering method that uses the Pearson Correlation Coefficient to calculate the similarity between users and then predict the missing QoS values based on the historical QoS records of similar users [22].

IPCC is the same as UPCC, except that it calculates the similarity between items [25].

UIPCC is a hybrid method proposed in [14] that combines the advantages of UPCC and IPCC methods to fully exploit the similarity between users and services for predicting missing QoS values.

WSRec this method is a hybrid approach that exploits both similar users and similar services, and linearly combines the prediction results of UPCC and IPCC [48].

PMF (*probabilistic matrix factorization*) is based on probabilistic matrix factorization, which introduces probability models to further improve matrix factorization models, and it also factorizes the user–service QoS matrix for the prediction [16].

NMF (*Non-negative Matrix Factorization*) this method applies non-negative matrix factorization to the user-item matrix to predict missing values without considering neighborhood information [11].

CAP (*Credibility-Aware Prediction*) CAP is a novel credibility-aware QoS prediction method that uses two-phase K-means clustering algorithms [35].

CMF (*Classic Matrix Factorization*) CMF is the classical matrix factorization method whose main objective is to builds a global model to make quality predictions based on the available quality information [9].

LFM (*Latent Factor Model*) LFM decomposes the user-service QoS matrix into a low-dimensional reduction to learn latent features of users and services and then predicts results based on the learned latent features [9].

SN-MF (*Service Neighbors-based MF*). SN-MF proposes three service-neighborhood enhanced prediction models for selecting neighbors of each service with the integration of latent features of service neighbors into the basic matrix factorization model [42].

[3] All parameters in the widely-used models are set to the same values as in the original papers.

LE-MF (*Linear-Ensemble MF*). LE-MF integrates context information of users and services respectively and trusts mechanism into traditional matrix factorization model to predict QoS values [39].

LR-MF (*Location and Reputation-aware MF*).LR-MF combines both the user's reputation and location information into the matrix factorization (MF) model to predict the missing QoS values [12].

NIMF (*Neighborhood-Integrated Matrix Factorization*). This method was the first one that integrates neighborhood-based information of users into the MF-based model to achieve higher quality predictions. It computes the PCCs to identify N(u) [50].

NAMF. This method also integrates users' neighborhood information to make quality predictions. Unlike NIMF, it identifies N(u) based on their geographical locations and the network map used to measure the network distance between users [32].

CSMF CSMF is a general context-sensitive matrix factorization method to make collaborative QoS predictions. By considering the complexity of service invocations, CSMF proposes to model user-to-service and environment-to-environment interactions simultaneously and fully exploit implicit and explicit contextual factors in QoS data [37].

JCM (*Joint CNN-MF*) is a new matrix factorization (MF) model which integrates a convolutional neural network (CNN). JCM is capable of learning deep latent features of a user or a service neighbor. JCM incorporates a novel similarity computation method to improve the accuracy of neighbor selection in edge computing environments.[43].

Table 6. Prediction accuracy comparison with different prediction methods in low data densities (*A smaller value means a better performance*)

Model	Training set density —— response time dataset							
	d = 5%		d = 10%		d = 15%		d = 20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UserMean	0.871	1.858	0.873	1.856	0.873	1.856	0.879	1.853
ItemMean	0.742	1.577	0.728	1.548	0.711	1.530	0.700	1.530
UPCC	0.955	2.126	0.782	1.856	0.671	1.726	0.597	1.717
IPCC	1. 102	2.258	0.878	1.989	0.784	1.862	0.722	1.794
UIPCC	0.847	1.920	0.729	1.730	0.612	1.590	0.552	1.587
CMF	0.611	1.414	0.516	1.356	0.491	1.216	0.459	1.198
NMF	0.618	1.574	0.604	1.549	0.599	1.534	0.598	1.533
PMF	0.567	1.473	0.499	1.286	0.472	1.216	0.449	1.182
NIMF	0.551	1.407	0.485	1.274	0.453	1.198	0.435	1.167
NAMF	0.538	1.385	0.485	1.259	0.452	1.207	0.435	1.144
CNMF	0.528	1.305	0.471	1.237	0.431	1.136	0.413	1.116
CAP	/	/	0.360	0.643	/	/	0.352	0.664
WSRec	0.679	1.488	0.621	1.426	0.603	1.368	0.602	1.351
LFM	0.578	1.501	0.564	1.320	0.543	1.271	0.535	1.226
SN-MF	0.631	1.396	0.537	1.269	0.500	1.226	0.487	1.207
LE-MF	0.573	1.370	0.513	1.251	0.482	1.204	0.464	1.180
LR-MF	0.551	1.415	0.478	1.257	0.446	1.212	0.434	1.142
JCM	0.513	1.332	0.466	1.250	0.450	1.185	0.436	1.180
Our	**0.292**	**0.399**	**0.153**	**0.214**	**0.120**	**0.169**	**0.101**	**0.142**

Table 7. Prediction accuracy comparison with different prediction methods in high data densities (*A smaller value means a better performance*)

Model	Training set density —— response time dataset							
	d = 30%		d = 50%		d = 70%		d = 90%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UserMean	0.868	1.835	0.877	1.859	0.877	1.866	0.872	1.841
ItemMean	0.682	1.529	0.674	1.504	0.674	1.518	0.680	1.526
UPCC	0.586	1.502	0.607	1.620	0.562	1.473	0.563	1.379
IPCC	0.631	1.472	0.586	1.618	0.543	1.470	0.668	1.417
CAP	0.331	0.678	/	/	0.228	0.581	0.188	0.582
WSRec	0.480	1.342	0.465	1.271	0.431	1.205	0.416	1.132
LFM	0.476	1.351	0.421	1.250	0.401	1.151	0.384	1.101
JCM	0.429	1.174	0.406	1.148	0.389	1.115	0.378	1.089
Our	**0.077**	**0.108**	**0.054**	**0.075**	**0.043**	**0.057**	**0.038**	**0.054**

5.4. Metrics

We evaluated the prediction accuracy of our model compared to other methods using the following metrics, including standard error metrics such as mean absolute error (*MAE*) and root mean square error (*RMSE*).

Mean Absolute Error (*MAE*): MAE is a quantity used to measure the prediction accuracy of QoS prediction methods; it indicates the average of the absolute difference between the predicted QoS value and the real QoS value of the service invoked by a user over the test records.

MAE is defined as follows.

$$MAE = \frac{\sum_{(u,i)\varepsilon M_t} \left| q_{u,i} - \hat{q}_{u,i} \right|}{\left| N(M_t) \right|} \tag{5}$$

Root Mean Square Error (*RMSE*): During the calculation of RMSE, the individual differences between the predicted values and the corresponding observed values are each squared and then averaged over the sample. Then the square root of the average is taken as the final result.

RMSE is defined as follows.

$$RMSE = \sqrt{\frac{\sum_{(u,i)\varepsilon M_t} (q_{u,i} - \hat{q}_{u,i})^2}{\left| N(M_t) \right|}} \tag{6}$$

Where M_t denotes the test set matrix and $N(M_t)$ is the number of QoS values in the test set M_t , $q_{u,i}$ and $q_{u,i}$ represent the real QoS value and the predicted QoS value, respectively. According to the above two definitions, MAE and RMSE both vary in (*0,* ∞), and a smaller value of them means higher prediction accuracy. They are widely used in the field of web service prediction.

5.5. Analyze of Our Results

In the following sections, we conduct a series of experiments to evaluate our model on different training sets with different densities (*5%, 10%, 15%, 20%, 30%, 50%, 70%, 90%*) and we investigate how the density key affects the prediction accuracy. The experiments are conducted on the response time dataset.

Evaluation of the Tables of Comparison From Table 6 and Table 7 and in comparison with the state-of-the-art and other methods, we can make the following observations:

- Our Auto-NF model achieves the lowest errors for both MAE and RMSE metrics in all cases of data densities, including low and high data densities. This indicates that our method can be applied to both sparse and dense datasets;
- Our Auto-NF model performs better prediction accuracy than all other prediction methods compared;
- Our Auto-NF model can better handle the data sparsity problem.

Impact of matrix density Fig. 3 shows the influence of density on prediction accuracy using MAE and RMSE errors. This figure shows that as the data density increases, both MAE and RMSE values decrease rapidly at first. However, they are still the same after the density increases. Therefore, our model produces the lowest prediction errors and has the best results in high density. The reason is that the more the data is trained, the better the quality of learning deep features becomes.

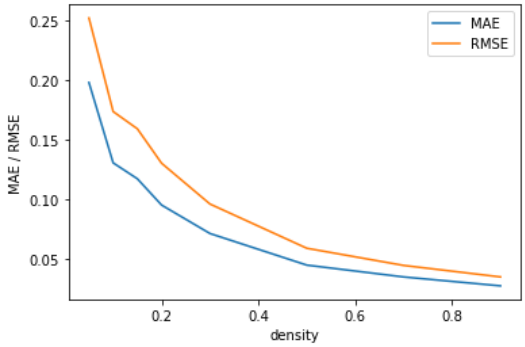

Fig. 3. The MAE and RMSE variation under different densities

Data validation To check if our model is sensitive to the overfitting problem, we needed to follow the evolution of the training and validation graphs based on MAE and RMSE metrics in the learning step and then compared their changes over time.
From Fig. 4 and Fig. 5, we can see that the gap between MAE and RMSE in the training and validation graphs is small, and when we attend a high density, the two graphs coincide, which means that our model is not overfitted.

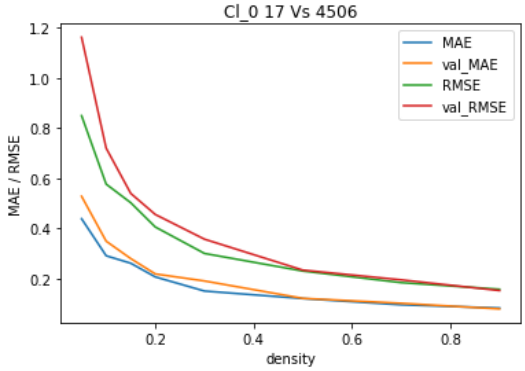

Fig. 4. The evolution of training and validation data based on MAE and RMSE errors in matrix 0

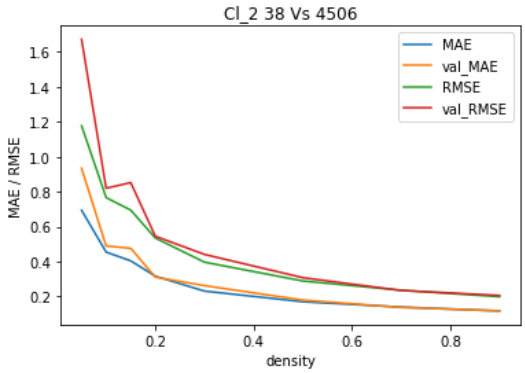

Fig. 5. The evolution of training and validation data based on MAE and RMSE errors in matrix 2

Prediction After the training step, first, we took the test set data from both the smallest and largest sub-matrices from our set of neighbor matrices as missing values and predicted their values with our autoencoder model. Then, we compared original and predicted values to show the difference between them and then evaluated the performance of our model.

For more explanation, we used MAE and RMSE measures to evaluate the predicted values compared to the real ones (*test set values*) and stored the difference between them as (*MAE, RMSE*) prediction error, as shown in Table 8 below.

Table 8. MAE and RMSE errors between some random real and predicted values for the evaluated model

Real value	7.489	1.519	2.570	3.768	1.389
Predicted value	6.991	1.433	2.255	3.570	1.024
MAE prediction error	0.26780975				
RMSE prediction error	0.3974242				
Real value	0	0.002	6.407	0	0.533
Predicted value	0.0009	-0.072	6.457	-0.037	0.548
MAE prediction error	0.04516501				
RMSE prediction error	0.05946906				

From Table 8, it can be seen that the predicted values are almost the same as the real ones in both sub-matrices (first, second, fifth, and sixth rows). There are small values for MAE and RMSE errors, especially when the actual and predicted values are close to zero (third, fourth, seventh, and eighth rows). This assures the performance and the high accuracy of our proposed model.

Real value is represented as:

$$realvalue = \int (predicted\, value) + (Error\, term\, predicton) \tag{7}$$

Where $\int$ is the activation function, the error term is MAE or RMSE used in this work.

Example MovieLens dataset [4]: is one of the well-known movie datasets that has been used for the evaluation of recommender systems. The numbers of users and movies in the Movielens dataset are 69878 and 10677, respectively. In this dataset, the users have provided ratings on a 5-star scale. Hence, based on the number of users and movies, this dataset includes 100, 000, 54 anonymous ratings.
For this example, we have selected 500 users and 5896 movies from the dataset Movielens.
We proceeded in the same way as for the WS-Dream dataset. We have got two clusters for neighbors for both (users and services). Each cluster is represented by a small matrix. Finally, we have obtained two sub-matrix of neighbors features.
Matrix 0: with a size of 9 users in the rows and 3069 movies in the cols.

[4] http://www.movieLens.org

Matrix 1:with a size of 495 users in the rows and 615 movies in the cols.
First, we replaced the missing QoS values in the sub-matrices once with zero and once with the average values. Then, we calculated the error values predicted for both sub-matrices at the highest and lowest densities. Finally, we compared between zero and average values, as shown in Table 9 follow:

Table 9. Comparison between zero and average QoS value based on RMSE and MAE errors (MovieLens dataset)

		Training set density —— MovieLens dataset							
		zero				average			
		10%		90%		10%		90%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Matrix 0	Prediction	0.0101	0.0154	0.0059	0.0078	0.0456	0.0640	0.0099	0.0142
Matrix 1	Prediction	0.4748	0.7549	0.3564	0.6092	0.2456	0.3917	0.2006	0.3472
average	Prediction	0.2424	0.3851	0.1811	0.3085	0.1456	0.2278	0.1052	0.1807

From Table 9 we can see that average gives the small error of the predicted values. So, the results are better when we replace the missing values with average than when we replace them with the zero values.

To make a new evaluation of our model, we compared the results of its last experiments on the movielens dataset with the previous ones on the WSDREAM dataset. The results are shown in Table 10 and Table 11.

Table 10. Prediction accuracy of our method compared between WSDREAM and MovieLens at low data density (*A smaller value means a better performance*)

Model	Training set density —— WSDREAM Vs MovieLens datasets							
	d = 5%		d = 10%		d = 15%		d = 20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Auto-NF WSDREAM	**0.292**	**0.399**	**0.153**	**0.214**	**0.120**	**0.169**	**0.101**	**0.142**
Auto-NF MovieLens	**0.153**	**0.236**	**0.144**	**0.227**	**0.140**	**0.222**	**0.137**	**0.218**

Table 11. Prediction accuracy of our method compared between WSDREAM and MovieLens at high data densities (*A smaller value means a better performance*)

Model	Training set density —— WSDREAM Vs MovieLens dataset							
	d = 30%		d = 50%		d = 70%		d = 90%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Auto-NF WSDREAM	**0.077**	**0.108**	**0.054**	**0.075**	**0.043**	**0.057**	**0.038**	**0.054**
Auto-NF MovieLens	**0.125**	**0.203**	**0.116**	**0.191**	**0.108**	**0.181**	**0.107**	**0.183**

Table 10 and Table 11 show that our model evaluated with movielens gives good results. However, most values of the WSDREAM dataset are better than that of movielens (the errors values for both MAE and RMSE in WSDREAM are smaller than those in the movielens). As a result, movieLens provides lower prediction accuracy than the WSDREAM of our model, although it performs well.

6. Discussion

Our principal goal is to select optimum services and recommend them to users based on their Quality of Service, which is the most important criterion considered in recommending services. However, due to the large number of available services on the internet, it is really hard for a user to invoke all candidate services to acquire their QoS values and then make a final decision about the optimal one. Thus, predicting the QoS values of services is an indispensable task to finish service selection and recommendation.
In summary, in this study, we have proposed a model to predict the missing QoS values of services using WSDREAM, the most important dataset in the domain of web services. The experimental results in section. 5 confirm that our method has improved prediction accuracy and reduced susceptibility to overfitting.
Using the WSDREAM dataset of web services allowed us to train our predictor model correctly and with a good performance. However, using a different dataset could lead to lower accuracy. To verify that, we have tried our model with MovieLens, the well-known dataset in the field of movie recommendation see **"Example"** in the previous section. 5.5

Table 12 aims to present the difference between our work and the works proposed in the papers of Yin et al. (2019) [43] and Smahi et al. (2018) [29].
From the experimental results section. 5 and Table 12, we derive the following findings:

Users in the same country may not be neighbors. They may not call the same services, or they may have different QoS values for the same services. As shown in the Table 13 below:

From Table 13, we can observe that the QoS value (the response time) of u1 invoking s1 is very close to u29, even though they are not in the same country. Similarly, u0 and u2 from different countries intend to select the same service s5.
Since service s0 and service s1 are in the same network, their QoS values largely depend on the network distance among users and services.

6.1. Advantages of our model

- Has a simple structure (autoencoder with input-output and one hidden layer);
- Capable of reading deeply hidden features;
- Handle the data sparsity problem;
- Less sensitive to the overfitting problem;
- Achieves the lowest errors for both MAE and RMSE in all cases of data densities, including low and high data densities;
- Outperforms all previous methods in comparison (see tables 6 and 7);
- Performed better prediction accuracy;
- Our model has succeeded in minimizing the discrepancy between input and output data (see Table 8).

Table 12. Comparison between our work and the works presented in the papers of: Yin et al. (2019) [43] and Smahi et al. (2018) [29]

	Our work	Yin et al. (2019) [43]	Smahi et al. (2018) [29]
Similarity computation	We propose a simple new concept, i.e., $\lambda_{u,v}$ based on the common services invoked by both users. $\lambda_{u,v}$ is integrated into Euclidean distance to compute the similarity between two users.	Yin, Y et al propose two new concepts, i.e., IIFU (inverse invocation frequency of users) and IIFS (inverse invocation frequency of services). IIFU and IIFS are integrated into Euclidean distance to compute the similarity between two users or two services.	/
Neighbors selection based clustering	We divide the input dataset into a series of clusters to reduce the data sparsity based on the QoS values. Each cluster is represented with a reduced matrix that has fewer columns and rows concerning the initial dataset.	/	Smahi, M.I. et al divide the input dataset into a series of clusters to reduce the data sparsity based on the country ID or the provider ID. Each cluster is represented with a reduced matrix that has fewer columns and rows concerning the initial dataset.
Data density values	5%, 10%, 15%, 20%, 30%, 50%, 70%, 90%	5%, 10%, 15%, 20%, 30%, 50%, 70%, 90%	80%
Over-fitting	In the learning step, we try with different sizes of hidden layer. Finally we find it 2048 neurons.	/	In the learning step, Smahi, et al perform a cross-validation to infer the best hidden layer size. Finally, they find it 120 neurons.
MAE and RMSE errors	Our model achieves the lowest errors for both MAE and RMSE in all cases of data densities, including low and high data densities. See tables 6 and 7 ”our” model.	See tables 6 and 7 ”JCM” model.	For 80% density and 120 neurons: MAE: 0.681 RMSE:1.369 See original paper.
Dataset	WS-DREAM Movielens ml10M	WS-DREAM	WS-DREAM
Technique	Autoencoder	CNN + MF	Autoencoder

Table 13. A capture from the real-world service invocation scenario rtdata from wsdream dataset

	S0 United State	S1 United State	S5 United State	S14 Argentina	S50 Australia
U0 United State	4.18	0.416	20.0	1.469	nan
U1 United State	1.166	0.335	20.0	0.653	0.771
U2 Japan	5.975	0.251	20.0	0.581	0.823
U29 United Kingdom	1.997	0.324	20.0	0.646	0.888
U30 Canada	1.638	2.408	20.0	10.755	4.132

6.2. Point limits of our model

– Using a different type of dataset like MovieLens could lead to lower accuracy.

7. Conclusion and Future Work

Due to the increasing number of services on the internet, it becomes harder to find the right ones. Furthermore, it is impracticable to check all services for their quality values since this consumes many resources. Therefore, users invoke a quite limited number of services, resulting in a sparse amount of data. Thus, a QoS prediction method is very important to find the most appropriate service among many functionally similar ones. In this work, we propose an effective model for predicting missing QoS values of services. We use historical QoS records for this purpose. Here we split the original dataset into partial matrices to eliminate sparsity and learn deep latent features to reduce the overfitting problem. Experimental results on a public dataset demonstrated that our model achieved the best results compared to the traditional and counterpart methods, both in low and high data densities. In contrast to other deep learning-based recommendation methods, that can achieve better results only in high data density. In the future, we plan to elaborate on another based-deep learning model, such as a convolutional neural network. Also, we will continue to improve our model.

Acknowledgments. The authors would like to acknowledge the ComSIS editor and reviewers for the thoughtful reading of this manuscript and for their relevant comments that helped us improve the quality of the paper.

References

1. Batmaz, Z., Yurekli, A., Bilge, A., Kaleli, C.: A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review 52(1), 1–37 (2019)
2. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363 (2013)
3. Chen, L., Ha, W.: Reliability prediction and qos selection for web service composition. International Journal of Computational Science and Engineering 16(2), 202–211 (2018)

4. Chen, S., Fan, Y., Tan, W., Zhang, J., Bai, B., Gao, Z.: Service recommendation based on separated time-aware collaborative poisson factorization. J. Web Eng. 16(7&8), 595–618 (2017)
5. Chen, S., Peng, Y., Mi, H., Wang, C., Huang, Z.: A cluster feature based approach for qos prediction in web service recommendation. In: 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). pp. 246–251. IEEE (2018)
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web. pp. 173–182 (2017)
7. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science 313(5786), 504–507 (2006)
8. Jin, Y., Wang, K., Zhang, Y., Yan, Y.: Neighborhood-aware web service quality prediction using deep learning. EURASIP Journal on Wireless Communications and Networking 2019(1), 1–10 (2019)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
10. Kuang, L., Gong, T., OuYang, S., Gao, H., Deng, S.: Offloading decision methods for multiple users with structured tasks in edge computing for smart cities. Future Generation Computer Systems 105, 717–729 (2020)
11. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
12. Li, S., Wen, J., Luo, F., Cheng, T., Xiong, Q.: A location and reputation aware matrix factorization approach for personalized quality of service prediction. In: 2017 IEEE International Conference on Web Services (ICWS). pp. 652–659. IEEE (2017)
13. Liang, H., Baldwin, T.: A probabilistic rating auto-encoder for personalized recommender systems. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. pp. 1863–1866 (2015)
14. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 39–46 (2007)
15. Mattiev, J., Kavšek, B.: Distance based clustering of class association rules to build a compact, accurate and descriptive classifier. Computer Science and Information Systems (00), 37–37 (2020)
16. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. Advances in neural information processing systems 20, 1257–1264 (2007)
17. Nilashi, M., Ibrahim, O., Bagherifard, K.: A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Systems with Applications 92, 507–520 (2018)
18. Papadakis, H., Panagiotakis, C., Fragopoulou, P.: Scor: a synthetic coordinate based recommender system. Expert Systems with Applications 79, 8–19 (2017)
19. Paradarami, T.K., Bastian, N.D., Wightman, J.L.: A hybrid recommender system using artificial neural networks. Expert Systems with Applications 83, 300–313 (2017)
20. Radovanović, S., Delibašić, B., Suknović, M.: Predicting dropout in online learning environments. Computer Science and Information Systems (00), 53–53 (2020)
21. Rama, K., Kumar, P., Bhasker, B.: Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution. Neural Computing and Applications pp. 1–11 (2021)
22. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work. pp. 175–186 (1994)
23. Rogić, S., Kašćelan, L.: Class balancing in customer segments classification using support vector machine rule extraction and ensemble learning. Computer Science and Information Systems (00), 52–52 (2020)

24. Rumelhart, D.E., McClelland, J.L., Group, P.R., et al.: Parallel distributed processing, vol. 1. IEEE Massachusetts (1988)
25. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp. 285–295 (2001)
26. Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H.: Personalized qos prediction forweb services via collaborative filtering. In: Ieee international conference on web services (icws 2007). pp. 439–446. IEEE (2007)
27. Shen, L., Pan, M., Liu, L., You, D., Li, F., Chen, Z.: Contexts enhance accuracy: On modeling context aware deep factorization machine for web api qos prediction. IEEE Access 8, 165551–165569 (2020)
28. Singla, P., Richardson, M.: Yes, there is a correlation: -from social networks to personal behavior on the web. In: Proceedings of the 17th international conference on World Wide Web. pp. 655–664 (2008)
29. Smahi, M.I., Hadjila, F., Tibermacine, C., Merzoug, M., Benamar, A.: An encoder-decoder architecture for the prediction of web service qos. In: European conference on service-oriented and cloud computing. pp. 74–89. Springer (2018)
30. Sun, H., Zheng, Z., Chen, J., Lyu, M.R.: Personalized web service recommendation via normal recovery collaborative filtering. IEEE Transactions on Services Computing 6(4), 573–579 (2012)
31. Tang, M., Jiang, Y., Liu, J., Liu, X.: Location-aware collaborative filtering for qos-based service recommendation. In: 2012 IEEE 19th international conference on web services. pp. 202–209. IEEE (2012)
32. Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., Zhang, T.: Collaborative web service quality prediction via exploiting matrix factorization and network map. IEEE Transactions on Network and Service Management 13(1), 126–137 (2016)
33. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A., Bottou, L.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research 11(12) (2010)
34. Wang, S., Zhao, Y., Huang, L., Xu, J., Hsu, C.H.: Qos prediction for service recommendations in mobile edge computing. Journal of Parallel and Distributed Computing 127, 134–144 (2019)
35. Wu, C., Qiu, W., Zheng, Z., Wang, X., Yang, X.: Qos prediction of web services based on two-phase k-means clustering. In: 2015 ieee international conference on web services. pp. 161–168. IEEE (2015)
36. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019)
37. Wu, H., Yue, K., Li, B., Zhang, B., Hsu, C.H.: Collaborative qos prediction with context-sensitive matrix factorization. Future Generation Computer Systems 82, 669–678 (2018)
38. Wu, H., Zhang, Z., Luo, J., Yue, K., Hsu, C.H.: Multiple attributes qos prediction via deep neural model with contexts. IEEE Transactions on Services Computing (2018)
39. Xu, Y., Yin, J., Deng, S., Xiong, N.N., Huang, J.: Context-aware qos prediction for web service recommendation and selection. Expert Systems with Applications 53, 75–86 (2016)
40. Xu, Y., Yin, J., Lo, W., Wu, Z.: Personalized location-aware qos prediction for web services using probabilistic matrix factorization. In: International Conference on Web Information Systems Engineering. pp. 229–242. Springer (2013)
41. Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 114–121 (2005)
42. Yin, J., Xu, Y.: Personalised qos–based web service recommendation with service neighbourhood–enhanced matrix factorisation. International Journal of Web and Grid Services 11(1), 39–56 (2015)

43. Yin, Y., Chen, L., Xu, Y., Wan, J., Zhang, H., Mai, Z.: Qos prediction for service recommendation with deep feature learning in edge computing environment. Mobile Networks and Applications pp. 1–11 (2019)
44. Yin, Y., Xu, Y., Xu, W., Gao, M., Yu, L., Pei, Y.: Collaborative service selection via ensemble learning in mixed mobile network environments. Entropy 19(7), 358 (2017)
45. Yin, Y., Zhang, W., Xu, Y., Zhang, H., Mai, Z., Yu, L.: Qos prediction for mobile edge service recommendation with auto-encoder. IEEE Access 7, 62312–62324 (2019)
46. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR) 52(1), 1–38 (2019)
47. Zhang, Y., Yin, C., Wu, Q., He, Q., Zhu, H.: Location-aware deep collaborative filtering for service recommendation. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019)
48. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: 2009 IEEE International Conference on Web Services. pp. 437–444. IEEE (2009)
49. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Transactions on services computing (2010)
50. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service qos prediction via neighborhood integrated matrix factorization. IEEE Transactions on Services Computing 6(3), 289–299 (2012)
51. Zhu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Carp: context-aware reliability prediction of black-box web services. In: 2017 IEEE International Conference on Web Services (ICWS). pp. 17–24. IEEE (2017)
52. Zhuang, F., Zhang, Z., Qian, M., Shi, C., Xie, X., He, Q.: Representation learning via dual-autoencoder for recommendation. Neural Networks 90, 83–89 (2017)

Fatima Zohra Merabet is a Ph.D. student at faculty of new information and communication technologies, University of Constantine 2 - Abdelhamid Mehri and affiliated to the LIRE laboratory. He received a license and Master's degrees in Information Systems and web technology in respectively 2016 and 2018 from the same faculty. His research interests revolve around: self-adaptive system, autoencoder, IoT, QoS prediction.

Djamel Benmerzoug is currently a full professor in the department of TLSI, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria. He holds a PhD in Computer science from Pierre & Marie Curie University (Paris - France). Pr Djamel Benmerzoug has published many articles in many International Conferences and Journals. He supervises many PhD and Master students. His current research interests include Internet of Things, Cloud Computing, Advanced Enterprises Systems, Multiagent Systems, Service Oriented Computing, and Business Processes Modelling and Verification.

Received: May 18, 2021; Accepted: October 05, 2021.