Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 9240843, 16 pages
https://doi.org/10.1155/2022/9240843

Research Article

Relationship Discovery and Hierarchical Embedding for Web Service Quality Prediction

Hualong Bu,[1] **Jing Xia**,[2] **Qilin Wu**,[1] **and Liping Chen**[1]

[1]*School of Information Engineering, Chaohu University, Hefei 238000, China*
[2]*School of Mathematics and Statistics, Chaohu University, Hefei 238000, China*

Correspondence should be addressed to Qilin Wu; lingqiw@126.com

Received 20 April 2022; Accepted 24 September 2022; Published 5 October 2022

Academic Editor: Ripon Chakrabortty

Web Services Quality Prediction has become a popular research theme in Cloud Computing and the Internet of Things. Graph Convolutional Network (GCN)-based methods are more efficient by aggregating feature information from the local graph neighborhood. Despite the fact that these prior works have demonstrated better prediction performance, they are still challenged as follows: (1) first, the user-service bipartite graph is essentially a heterogeneous graph that contains four kinds of relationships. Previous GCN-based models have only focused on using some of these relationships. Therefore, how to fully mine and use the above relationships is critical to improving the prediction accuracy. (2) After the embedding is obtained from the GCNs, the commonly used similarity calculation methods for downstream prediction need to traverse the data one by one, which is time-consuming. To address these challenges, this work proposes a novel relationship discovery and hierarchical embedding method based on GCNs (named as RDHE), which designs a dual mechanism to represent services and users, respectively, designs a new community discovery method and a fast similarity calculation process, which can fully mine and utilize the relationships in the graph. The results of the experiment on the real data set show that this method greatly improved the accuracy of the web service quality prediction.

1. Introduction

More and more developers have paid attention to web services in the form of Web APIs [1]. By the end of March 22, 2022, ProgrammableWeb (the largest web service platform) has registered more than 24,417 APIs. Developers can access services through simple APIs, which provide a great convenience for development. As the number of services is increasing, users have only called a small number of services in reality, resulting in most service qualities being unknown in advance. Therefore, how to appropriately predict the quality of the web service becomes a serious issue [2].

Accompanied by the rising of deep learning technology, various deep web service quality prediction models have been continuously proposed [3–5]. Deep learning methods such as CNN and RNN can extract hidden information that cannot be treated by traditional matrix decomposition technology [6, 7] to improve the accuracy, but these methods do not take the topology information into account at all. In recent years, numerous graph convolutional networks (GCNs) have been proposed [8–12]. GCNs can integrate feature information from local graph neighborhood, which have been demonstrated to be powerful for graphical representation [13–15]. Neural Graph Collaborative Filtering (NGCF) [13] exploits the user-service graph structure and integrates the users and services interactions into the embedding process. PinSage [14] designs an efficient random walk method with graph convolutions to generate node embedding that incorporates both graph structure and node information. LDNM [15] transforms each service document into feature vectors by using LDA and Doc2vec, then applies Node2vec and MLP neural network for Web Service Classification.

Although GCNs have achieved great achievements in web service quality prediction, we argue that there are still suffering from the following challenges.

First, the user-service bipartite graph is essentially a heterogeneous graph. As shown in Figure 1, the bipartite user-service graph contains four types of relationships such as (1) user-user shared-invoking relationship; (2) user-user potential relationship; (3) user-server relationship; and (4) server-server relationship.

Taking the user node **u1** as an example, the first-order neighborhood of user **u1** contains the service nodes **s1** and **s2** which indicates the user-server relationship, and the second-order neighborhood contains the user node **u2** that has called the service **s2** before, which indicates the user-user shared-invoking relationship.

Taking the service node **s1** as an example, the first-order neighborhood of **s1** contains the user nodes **s1**, and the second-order neighborhood contains the service node **s2**. Node **s1** and node **s2** form the server-server relationship.

Furthermore, taking user **u1**, user **u5**, and user **u6** as an example, these users share characteristics (e.g., they may have common interests), but they have not revoked the same service in the past. In this way, users **u1**, **u5**, and **u6** form a user-user potential relationship with each other. Generally speaking, without prior domain knowledge, the user-user potential relationships are unknown in advance.

Previous GCN-based models have only focused on using one or more of these relationships. Therefore, how to fully mine the above relationships and make use of these relationships is critical to improve the prediction accuracy.

Second, when aggregating node features with the help of topology information, traditional methods must calculate the similarity. The existing similarity calculation research mainly focuses on how to improve the accuracy of downstream tasks, and it is necessary to traverse all samples one by one during traversal [16, 17]. Obviously, similarity calculation is a very time-consuming process when the number of samples is large. In order to reduce the computational complexity, a more efficient similarity calculation method needs to be adopted.

In addition, we notice that the web service quality prediction methods based on GCN have also faced the matrix sparsity problem to be solved. To address the aforementioned challenges simultaneously, this article proposes a novel relationship discovery and hierarchical embedding method (RDHE) based on GCNs for web service quality prediction, which can fully utilize four kinds of relationships to achieve better prediction accuracy. Given a user-service bipartite graph or QoS matrix, we first reconstruct the user-neighborhood graph and the service-neighborhood graph, respectively. Second, we propagate and aggregate the features of nodes from their local neighborhood by user embedding module and service embedding module, respectively. Third, we cluster the user embedding and service embedding with the help of clustering component. In addition, we conduct the communities using the community discovery module to instruct the downstream clustering. After these, we fuse the user-based and server-based predictions for the ultimate prediction. The experimental results show that our RDHE method can greatly improve the accuracy. The main contributions of this work are summarized as follows.

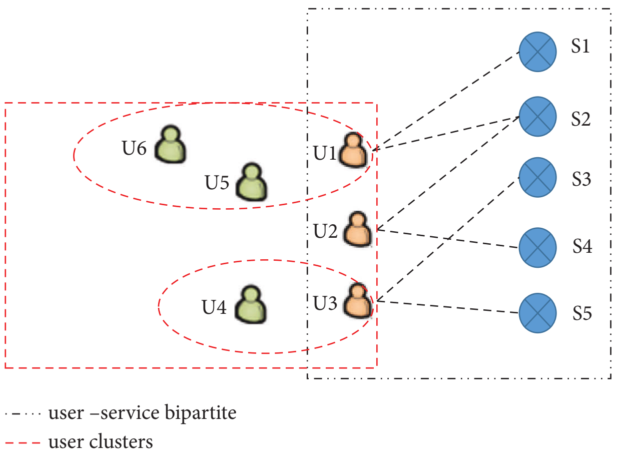

Figure 1: An illustration of the user and service relationships.

(1) A dual model to effectively learn neighborhood embedding from the user perspective and service perspective independently, which can lighten the damage caused by sparsity.
(2) Two community discovery methods are proposed to instruct the user- and service-based predictions to improve the accuracy of the final prediction, which can fully mine and use the above four kinds of relationship.
(3) A fast similarity calculation method to handle large-scale data, which can avoid the curse of dimensionality for downstream clustering tasks.
(4) A large number of experiments are carried out on the real WS-Dream data set. The experimental results show that compared with the classic methods, the prediction precision of our RDHE method improves a lot.

The remainder of this article is organized as follows: Section 2 introduces the related work for Web Service Quality prediction. Section 3 shows the whole framework. Section 4 explains in detail the implementation process of the RDHE algorithm. Section 5 presents the experimental results and analysis. We summarize the full text in the Conclusion section.

2. Related Work

2.1. Deep Learning Models for Embedding. With the development of cloud computing and the Internet of things, various web services are registered and invoked on the network, and the prediction of web service quality has become a popular research direction. Our work builds on some recent advances in graph neural network representation learning methods.

The concept of neural networks for graph data was first outlined by Gori et al. [18] and further elaborated by Scarselli et al. [19]. However, these initial methods required running expensive neural “message passing” algorithms to converge and were constrained by the scale of graph data, which was further improved by gated graph sequential neural networks [20] to graphs with <10,000 nodes, but the cost is still high.

A large number of approaches to "graph convolution" or graph convolutional networks (GCNs) have recently emerged, originating from the work of Bruna et al., which developed a graph convolution based on spectral graph theory [21]. Since then, many authors have proposed improvements, extensions, and approximations to these spectral convolutions [8–12, 22, 23], and achieved good results on tasks such as node classification and link prediction. These methods have consistently outperformed matrix factorization or random walk-based techniques (e.g., node2vec [24] and DeepWalk [25]), and their success has led to a surge of interest in applying GCN-based methods to applications such as recommender systems, of which Hamilton et al. [26] and Bronstein et al. [27] conducted a comprehensive survey of recent progress.

However, despite the success of GCN algorithms, the above works are mainly based on homogeneous graphs, ignoring the heterogeneous structure effects of graphs. Since the representation learning of heterogeneous graph neural network has the strong expressive ability and the characteristics of effectively combining node attribute features and structural information, it can not only solve the problems of network data such as data sparse [28], but also has achieved remarkable results in various downstream tasks, such as node classification [29], link prediction [30], node clustering [31, 32], and recommendation [33–35].

We found that the heterogeneity of the graph structure causes different node types to have different semantic features, and even if the node types are the same, there may be different community attributes (e.g., although they are all user nodes, their implied points of interest are different). Inspired by the community discovery algorithm [36], our work introduces community discovery (further discovering distinct communities of cohorts to guide downstream work) into graph neural network representation learning.

2.2. Similarity Calculation. Our method is also relevant to the field of similarity calculation. Similarity calculation indicators can evaluate the similarity between data, and provide an accurate and effective basis for data analysis. Previous studies on similarity calculation mainly focus on the following aspects:

(1) The design of basic measurement indicators, which can be divided into distance-based measurement methods and similarity coefficient-based measurement methods [37, 38]. Distance-based measurement methods mainly include Euclidean distance, Manhattan distance, etc., while similarity coefficient-based measurement methods mainly include Cosine similarity, Pearson correlation coefficient, etc. Among them, the most common methods are the application of Euclidean distance and Cosine similarity.

(2) Embedding similarity strategy design, this type of method mainly studies how to improve the guidance of similarity calculation for downstream tasks [16], such as ontology-based methods, semantic-based methods, and some other hybrid methods [17].

The above methods mainly consider the effectiveness or learning accuracy of the similarity measurement method itself, but due to the large amount of current data, the time performance of similarity calculation is insufficient. To this end, this article proposes a similarity calculation method based on a neural network. The main idea is to reduce the time required for similarity calculation as much as possible under the premise of ensuring accuracy.

3. The Whole Framework

We will briefly describe the framework of our proposed method RDHE, as shown in Figure 2. The entire framework consists of six components:

(i) User embedding module, which contains the reconstruction of the User Neighborhood component, Wide and Deep Embedding component, and GraphSAGE Embedding components.

(ii) Service embedding module, which contains similar components as user embedding module.

(iii) User-community discovery module, in which a new community discovery method is proposed to instruct the service-based clustering.

(iv) User clustering module, in which a clustering method is proposed for user clustering.

(v) Service clustering module, in which we get the service cluster and the user-cluster affiliation degree matrix.

(vi) At last, a fusion operation is used to fuse the user-based prediction and server-based prediction.

4. The Proposed Prediction Method

4.1. Reconstruction of Neighborhood Graph. Given a QoS matrix $Q^{m\times n}$, m and n are the number of users and services, respectively. $X^{(m+n)\times d}$ is the feature matrix and d is the dimension of features. The corresponding bipartite graph $G(U,S,E,X)$ can be simply derived from $Q^{m\times n}$. The bipartite graph $G(U,S,E,X)$ is a typical heterogeneous graph, the direct neighbors of users are the service nodes and the direct neighbors of services are the user nodes.

Assume that it is more efficient to learn different neighborhood embedding from the user perspective and service perspective independently, rather than recursively updating neighborhood embedding regardless of the node types. Based on this, we reconstruct a user-neighborhood graph and a service-neighborhood graph from the $G(U,S,E,X)$ or its corresponding QoS matrix directly. If user u_1 and user u_2 are connected to the same service, they become direct neighbors in the user-neighborhood graph. In other words, if the elements of the i-th column in the QoS matrix are nonzero, then corresponding users of service i become direct neighbors in the user-neighborhood graph.

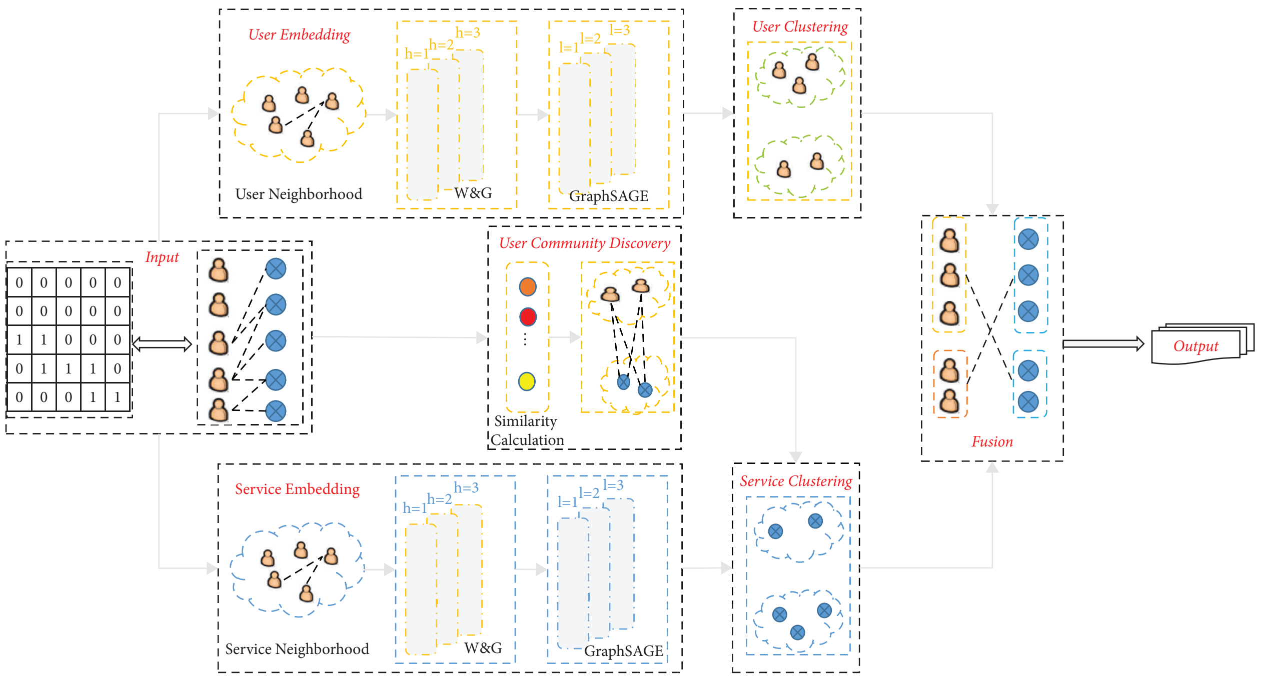

FIGURE 2: The framework of web service quality prediction. The input can be QoS matrix or corresponding bipartite graph. The output is the prediction of missing QoS values.

Likewise, if two services connect to the same user (the corresponding row's elements are nonzero), they become direct neighbors in the service-neighborhood graph. In this way, we can reconstruct the user-neighborhood graph $G(U, E', X_u)$, $X_u \in \mathbb{R}^{m\times d}$ and service-neighborhood graph $G'(S, E'', X_s)$, $X_s \in \mathbb{R}^{n\times d}$.

It should be noted that $G(U, E', X_u)$ contains two kinds of user relationship information, one is the user-user shared-invoking relationship which is constructed by the above method and is reflected in the topological structure of $G(U, E', X_s)$; the other is the user-user potential relationship which contains the latent community-sharing information, and we will discover it through the community discovery module (more details will be discussed in Section 4.3).

4.2. Sparsity and Large-Scale Handling. In general, the feature matrixes of X_u and X_s are sparse and high-rank, which will lead to the failure of methods which directly based on the feature matrixes. Here, we adopt the Wide & Deep [39] learning model to better represent sparse features by simultaneously training a linear LDA model and a neural network model.

Given a user-neighborhood graph $G(U, E', X_u)$ and its nodes feature matrix X_u, we can get the representation as,

$$E_u^w = WD_u X_u, \tag{1}$$

where $WD_u(\bullet)$ is a Wide & Deep learning module for users. Similarly, the service representation can be obtained by the Wide & Deep learning module as

$$E_s^w = WD_s X_s. \tag{2}$$

We then output E_u^w and E_s^w as inputs of GraphSAGE [40], which is a general framework that takes advantage of the node feature to efficiently embed large-scale data, and can generate embedding by sampling and aggregating features from a node's neighborhood as follows:

$$\begin{aligned}
H_{\mathcal{N}_u}^t &= \text{aggregate}_u^t\left(\left\{H_i^{t-1}, i \in \mathcal{N}_u\right\}\right),\\
H_u^t &= \sigma\left(W_u^t \bullet \text{concat}\left(H_u^{t-1}, H_{\mathcal{N}_u}^t\right)\right),\\
H_{\mathcal{N}_s}^t &= \text{aggregate}_s^t\left(\left\{H_j^{t-1}, j \in \mathcal{N}_s\right\}\right),\\
H_s^t &= \sigma\left(W_s^t \bullet \text{concat}\left(H_s^{t-1}, H_{\mathcal{N}_s}^t\right)\right),
\end{aligned} \tag{3}$$

where $\text{aggregate}_u^t(\bullet)$ and $\text{aggregate}_s^t(\bullet)$ are aggregating functions, σ is the activate function, W_u^t and W_s^t are the parameter matrixes which can be trained. $H_s^0 = E_s^w$ and $H_u^0 = E_u^w$ as original inputs.

Note that concat$(\bullet)$ is referring to the layer-aggregation mechanism. Since concatenation does not require additional parameters to learn, it has been shown quite simply and effectively for graph neural networks. In addition to concatenation, other strategies such as max pooling, can also be applied. Here, we adapt the concatenation for its simplicity.

4.3. Clustering with the Instruction of Community Discovery

4.3.1. Fast Similarity Calculation Process. Clustering divides a data set into different clusters according to a certain distance criterion so that the similarity of samples in the same cluster is as large as possible, and the difference of samples not in the same cluster is also as large as possible.

Traditionally, a clustering algorithm usually only needs a similarity calculation function $\mathrm{sim}(s_i, s_j)$ to get started. Traditional similar calculation process needs to traverse the embedding one by one when calculating $\mathrm{sim}(s_i, s_j)$, which are suitable for small-scale data sets, but the operating time will increase exponentially with the expansion of the dataset size [38]. In this work, a neural network-based similarity calculation process is designed for downstream similarity calculation and reduced the time complexity as shown in Figure 3.

In Figure 3, in order to improve the similarity calculation performance, a network model that can fit the similarity mapping relationship of the dataset is established. BP neural network has excellent nonlinear approximation, self-learning, self-adaptation, and generalization ability. According to the universal approximation theorem, if a feedforward neural network has a linear output layer and at least one hidden layer with an activation function, as long as there are enough hidden neurons, it can approximate any continuous function in a finite-dimensional space with arbitrary precision. In actual use, the computational time and space overhead must be considered, the exact representation is usually abandoned, and the nonlinear mapping relationship between the dataset and the label set is approximated by finding suitable parameters on the basis of the approximate representation. This provides a theoretical basis for using BP neural network for fast similarity calculation.

After the similarity calculation formula is selected, the numerical value of the data and the similarity between the data can constitute a definite multi-input and multi-output nonlinear mapping relationship, so the BP neural network can be used to fit this mapping relationship. In order to reduce the time of traversal calculation, the model is trained based on the similarity of some samples, and the accuracy is exchanged for speed, and the nonlinear similarity mapping relationship of the complete data set is represented within the allowable error range, so as to calculate the approximate similarity of the complete data set.

We give a brief complexity analysis here. Set $X \in R^{n\times m}$. For tradition calculating method, the total time complexity is $O(mn^2)$. The time complexity of the neural network method can be divided into precise calculation part and network part. In the precise calculation part, let b be the sampling ratio, $0 < b < 0.5$, the precise calculation complexity is $O(b^2 mn^2)$. Assuming the number of the inputted neurons per hidden layer is m, the hidden layer number is t, and the number of output neurons is 1, the time complexity of the neural network method is $O((1 - \mathrm{b})mnt)$. Generally, $t \ll n$, so the neural network method is more efficient than the traversal calculation method.

4.3.2. User Clustering.

4.3.2. User Clustering. We note the cluster center vector as the mean vector of the feature vectors of all the users in the cluster, and calculate the similarity between the feature vector of the user u in cluster i and the rating vector of cluster i as the user u's interest in the cluster i.

The similarity calculation function is the widely used Pearson Correlation Coefficient (PCC). We use PCC to measure the similarities between vector r_i and r_i, and get the user clusters UC on the user embedding H_u^t. We use the function $f(x) = (x + 1)/2$ to project the PCC into range [0, 1]. The PCC is defined as:

$$\mathrm{sim}(i, j) = \frac{\sum_{p\in P}\left(x_{ip} - \overline{x}_i\right)\left(x_{jp} - \overline{x}_j\right)}{\sqrt{\sum_{p\in P}\left(x_{ip} - \overline{x}_i\right)^2\left(x_{jp} - \overline{x}_j\right)^2}}, \tag{4}$$

where P is the interaction of the two vectors.

Based on the assumption that users with more social relationships are more representative, we can derive the weighted cluster center vectors.

$$WC_i = \frac{\sum_{g\in UC(i)}\left|relation_i(g)\right| r_{u_g}}{\sum_{g\in UC(i)}\left|UC(i)\right|}, \tag{5}$$

where $C(i)$ represents the set of users in cluster i, $\mathrm{relation}_i(g)$ represents the set of users who have a direct relationship with user u_g in cluster $i.r_{u_g}$ is the feature vector of user u_g.

Finally, we can obtain the user clusters UC and the rating matrix UCR by formula (4) and (5).

4.3.3. Service Clustering with the Instruction of Community Discovery. Finding and utilizing community information on the original input QoS matrix without prior knowledge can be connected to the concept of community discovery. A fuzzy clustering algorithm based on interest preference is proposed here. The algorithm uses the user's behavior records and the clusters to which the services belong to find a set of users similar to the target user's interest preference.

Given the original input QoS matrix $Q^{m\times n}$, the users' ratings on services form a $m \times n$ rating matrix $R = [r_{ij}]$, where r_{ij} is the rating of user u_i on service s_i, r_i is the rating vector of user u_i. We note the community rating vector as the mean vector of the rating vectors of all the users in the community.

First, we use the PCC to calculate the similarity of the service embedding H_s^t to get the service clusters SC.

Next, we derive the distribution of clusters SC which have been rated by u_i can be described as:

$$\mathrm{Dist}(i) = \left(\frac{\left|S_i^{SC_1}\right|}{\left|S_i\right|}, \frac{\left|S_i^{SC_2}\right|}{\left|S_i\right|}, \ldots, \frac{\left|S_i^{SC_p}\right|}{\left|S_i\right|}\right), \tag{6}$$

where $S_i$represents the set of services rated by user u_i, and $S_i^{SC_k}$ represents the set of services rated by user $u_i$in cluster C_k.

Next, due to the sparse data of the user-service rating matrix, the user category preference vector also has a sparseness problem, we define a more effective similarity function than traditional Minkowski distance as follows:

$$\mathrm{sim}\left(Dist(i), g_j\right) = \frac{1}{\tau_i}\sum_{k=1}^{p}\tau_{ik}\left(Dist(i)_k, g_{j,k}\right), \tag{7}$$

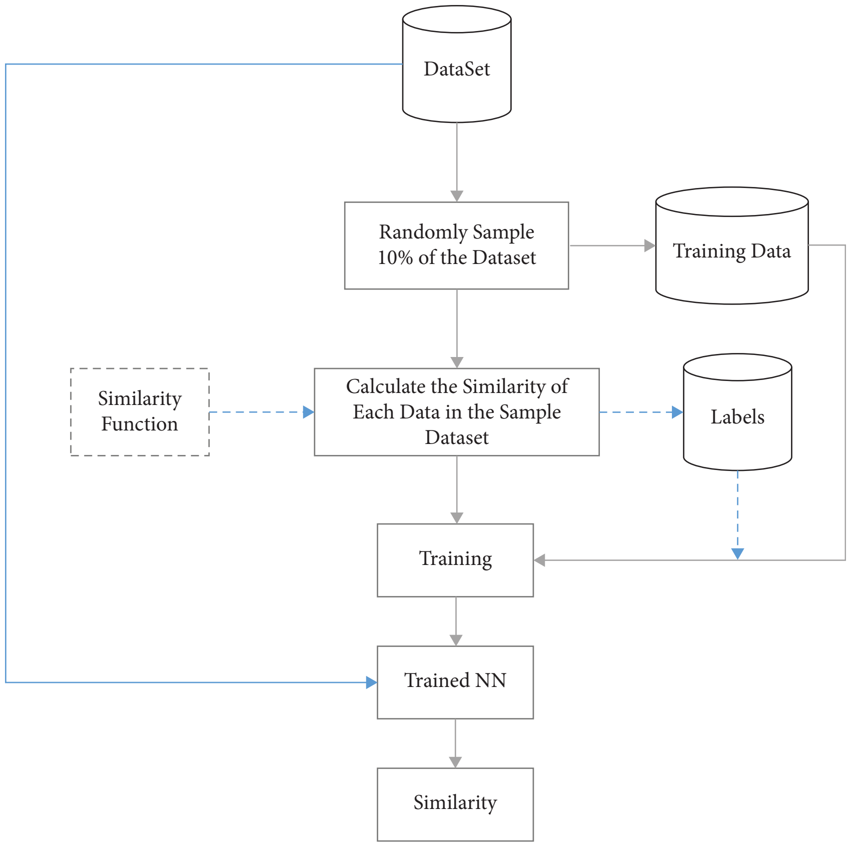
DataSet
Randomly Sample 10% of the Dataset
Training Data
Similarity Function
Calculate the Similarity of Each Data in the Sample Dataset
Labels
Training
Trained NN
Similarity

Figure 3: Fast similarity calculation.

where $\tau_{ik} = \begin{cases} 1, Dist(i)_k \neq 0 \\ 0, Dist(i)_k = 0 \end{cases}$, $\tau_i = \sum_{k=1}^{p} \tau_{ik}$. l is the number of clusters, g_j represents a certain cluster center vector, $j = 1, 2, \ldots, l$. $g_{j,k}$ is the k-th element in g_j.

Then we derive the objective function:

$$\text{Objective}(USA, G) = \sum_{i=1}^{m} \sum_{j=1}^{l} u_{ij}^2 \text{sim}\left(Dist(i), g_j\right), \tag{8}$$

where $u_{ij} \in [0, 1]$ represents the affiliation degree of user u_i to cluster C_j and $\sum_{k=1}^{l} u_{ij} = 1$. $USA = u_{ij}^{m\times l}$ represents the user-cluster affiliation degree matrix. $G = (g_1, g_2, \ldots, g_l)^T$ represents the center cluster matrix.

Finally, we update the values of the user-cluster affiliation matrix USA and cluster center matrix G through iterative operation, and gradually reduce the error of the objective function to the pre-set convergence threshold and terminate the iteration.

4.4. Joint Prediction. In order to reduce the influence of the redundant neighbors on the prediction, we predict the missing QoS values $P_u(i, j)$ of user j as follows:

$$P_u(i, j) = \frac{\sum_{i=1}^{\text{top}-k} \left(qos_{u_i} \times UCR_{it}\right)}{\text{top} - k}, \tag{9}$$

where u_i is the i-th user in the cluster UC_t which user u_j belongs to, qos_{u_i} represents the QoS value of user u_i, UCR_{it} means the rating score of user u_i to cluster UC_t, and top $- k$ represents the number of how many neighbors to be selected.

The formula for predicting missing values $P_s(i, j)$ of service j based on the service clustering result is as follows:

$$P_s(i, j) = \frac{\sum_{i=1}^{top-k} \left(qos_{u_i} \times US_{it}\right)}{\text{top} - k}, \tag{10}$$

where u_i is the i-th affiliated user of the server cluster SC_t which user u_j affiliated to, qos_{u_i} represents the QoS value of user u_i, US_{it} means the affiliation degree of user u_i to cluster SC_t, and top $- k$ represents the number of how many neighbors to be selected.

To fully consider the similarity of similar users and similar services, this article combines formulas (9) and (10) to propose a hybrid prediction method as follows:

$$P_{u,s}(i, j) = \lambda \times P_u(i, j) + (1 - \lambda) \times P_s(i, j). \tag{11}$$

The parameter $\lambda \in [0, 1]$ indicates that the predicted value depends on the proportion of similar users and similar services. When $\lambda = 1$, only the user-based method is used for prediction; when $\lambda = 0$, only the service-based method is used for prediction. If user u does not call any service that has been used or service s has not been called by any user, the average of all service QoS values will be used for prediction.

4.5. Model Training. To optimize the model parameters of RDHE, we need to optimize an objective function. In the training process, we denote S^+ as positive samples and denote S^- as negative samples, respectively. Given user $u, y_{u,s}$ is the ground truth label, if user u has invoked the service s, then $y_{u,s} = 1$, otherwise $y_{u,s} = 0$, $\hat{y}_{u,s}$ is the predicted label for service s. The loss function is derived as:

$$\% -\sum_{u}\sum_{i\in S^+\cup S^-}\left\{y_{u,s}\log P\left(\hat{y}_{u,s}=1\right)+\left(1-y_{u,s}\right)\log P\left(\hat{y}_{u,s}=0\right)\right\}. \tag{12}$$

Our model contains service embedding E_s^F and user embedding E_u^F, they are hierarchical trained and learned together. Since the raw features are very large and highly sparse for large-scale graphs, we do not use one-hot vectors to represent the nodes. By embedding high-dimensional sparse features into a low-dimensional latent space, the model can be easy to train. It is worth noting that, in terms of model size, the majority of the RDHE model parameters come from the embedding. We adopted the Cluster-GCN Mini-batch Trick [41] and Adam [42] optimization for GCNs.

Overfitting is another long-term problem for optimizing deep neural network models. The dropout strategy [43] is a common method to solve the overfitting of the neural network, which has the characteristics of simplicity and good effect. Here, we employ node and feature dropout strategies to prevent RDHE from overfitting. When the parameters, only part of them will be updated. Moreover, as a dropout is disabled during testing, the whole network is used for prediction.

5. Experiment

In this work, we performed a lot of experiments on real-world datasets to evaluate our RDHE method. We hope to answer the following questions:

(i) How does RDHE perform compared to the state-of-the-art methods?

(ii) How do different hyper-parameter settings affect RDHE?

5.1. Experimental Data and Evaluation Indicators. In our experiments, in order to evaluate the effectiveness of the prediction method proposed in this article, we use the WS-Dream dataset [44]. The two matrices in this dataset contain 1974675 response time (RT) and throughput (TP) data generated by 339 users calling 5825 services. There is a QoS record generated by invoking between each user and each Web service. The real recorded values of RT are between [0, 1].

To evaluate the effectiveness of service quality prediction, this work uses the mean absolute error (MAE) and the root mean square error (RMSE) indicators for test, which are defined as follows.

$$\begin{aligned}\text{MAE} &= \frac{\sum_{u,s}|r(u,s)-p(u,s)|}{N},\\ \text{RMSE} &= \sqrt{\frac{\sum_{u,s}\left(r(u,s)-p(u,s)\right)^2}{N}},\end{aligned} \tag{13}$$

where $r(u,s)$ and $p(u,s)$ represent the actual QoS and predicted QoS values, respectively, and N represents the number of predictions. It can be seen from the formula that the smaller the values of MAE and RMSE, the better the prediction performance, and the RMSE is more sensitive to the prediction error.

5.2. Baselines. We compare our proposed RDHE method with 10 state-of-the-art methods, which is described as follows:

(1) LDA [45]: LDA is an unsupervised topic modeling algorithm, which has been widely used in recommender systems. Here, each service is associated with a distribution on the topic of users, each user group is a distribution of users, and we use LDA to discover hidden relationships between users and services in the prediction task.

(2) Node2Vec [24]: Node2Vec is an improved version of DeepWalk that takes into account structure and homogeneity. In this method, we use the Node2vec model to maintain node neighbor information from the user- and service-neighborhood graphs for downstream tasks.

(3) GCN [8]: GCN is a classic semi-supervised model for extracting graph features. In this article, we treat both the services and users as nodes, then use the GCN to obtain the embedded representation of the graph.

(4) Wide & Deep [39]: in this method, we only use the Wide & Deep method to obtain user and service embedding for downstream similarity calculation and prediction tasks.

(5) GraphSAGE [40]: in this method, we only use the GraphSAGE module to obtain user and service feature representations for downstream tasks.

(6) NGCF [13]: in this method, we use the NGCF module to exploit the user-service graph structure by propagating high-order connectivity embedding on it; next the prediction task is imposed on it.

(7) PinSage [14]: in this method, we use a PinSage module to combine to generate embedding of nodes that incorporate both graph structure as well as node feature information. Then we perform similarity calculation and prediction tasks on it.

(8) KGAT [34]: in this method, we use a KGAT module to propagate the embedding by aggregating information from the constructed knowledge graph for prediction.

(9) GraphRec [12]: in this method, we use a GraphRec module to capture interactions and heterogeneous strengths in the user-service graph for prediction.

(10) NIA-GCN [35]: in this method, we adopt the NIA-GCN module to capture relationships between pairs of neighbors at each GCN layer for prediction.

LDNM is a fusion method of LDA, Doc2vec, and Node2vec, which have been adopted, respectively, thus we will not compare it.

5.3. Parameter Setting. We implement our method on the basis of Pytorch2 and use the Adam optimizer for all neural network-based methods. All the baselines and our algorithm are fully trained with up to 500 epochs, and the number of negative samples is 20. Moreover, we use validation set data for parameter tuning for all the methods.

In the real world, most of the users have only called a few services, therefore, in this work, to make the experiments more realistic, we randomly delete a certain number of QoS values from the initial RT and TP matrices to generate low-density matrices. For example, a matrix density of 5% means that we randomly select 5% of the QoS values to predict the remaining 95% of the QoS values. The removed QoS values are used as expected values to study prediction accuracy. Each training set data contain 10% of the validation set data; for example, 50% of the training set size means that 40% of the data are used for training and the remaining 10% of data are for validation.

The default values of the parameters and hyper-parameters of the baselines are shown in Table 1. If there are multiple datasets and the parameters are not specified, then the default parameters are the same. Specifically, we set the number of topics to 20 for all LDA-contained methods, where each latent topic corresponds to a specific service cluster. For our method, the Wide & Deep and GraphSAGE parameters are the same as in Table 1. We set the hidden layer the same as the embedding size and the activation function as ReLU. We used four hidden layers for the NN components of the Similarity Calculation and Prediction Module and its learning rate = 0.03, dropout = 0.5, and optimizable parameter $\delta = 0.6$. The effect of different values of λ, top-k, and other hyper-parameters will be shown in chapter 5.4.

5.4. Experiment Result. We first compare the prediction accuracy of different methods and then compare the effect of different parameters in RDHE, all the results and details will be discussed below.

5.4.1. Prediction Accuracy Comparison. In the experiment, the matrix density is set to 0.1–0.3 on the WS-Dream dataset (divided into RT and TP sub-datasets), and gradually increases at an interval of 0.05. We set $k = 3$ and $\lambda = 0.4$ to obtain the MAE and RMSE results. The experimental results are compared in Tables 2 and 3.

Tables 2 and 3 compare the prediction accuracy of 10 state-of-the-art methods with RDHD. It can be seen that our RDHE method has obtained better experimental results with both MAE and RMSE indicators, compared to all other baseline methods. Take the result of the MAE indicator as an example, the RDHE method is better than the four classic graph algorithms (Node2vec, GCN, Wide & Deep, and GraphSAGE) with 18.84%, 21.50%, 15.47%, and 12.50% achievements, respectively; it also achieves 11.11%, 9.19%, 9.19%, 5.62%, and 3.45% improvement than NGCF, PinSage, KGAT, GraphRec, and NIA-GCN, respectively. Similarly, take the result of the RMSE indicator in Table 3 as an example, the RDHE method achieved 14.48%, 16.85%, 14.19%, 13.74%, 12.33%, 8.51%, 7.98%, 11.21%, and 8.04% accuracy improvement than the nine GCNs methods based on the RMSE indicators.

Tables 2 and 3 also show the prediction accuracy changes of the different methods as the matrix density increases. As the matrix density increases, each method achieves better results than at lower densities, and the RDHE method improves even more.

5.4.2. The Influence of λ for Prediction Accuracy. As shown in (11), the parameter λ is a weight parameter, which determines the degree of dependence of the results based on the user's similarity and service's similarity. For example, if $\lambda = 1$, only the user-based method is used for prediction, and if $\lambda = 0$, only the service-based method is used for prediction. In order to study the influence of λ on the prediction results, the matrix density is set to 0.1, the number of similar users and the number of similar services is 6, and the λ ranges from 0.1 to 0.9 with an interval of 0.05. The experimental results are shown in Figure 4.

Among them, Figures 4(a) and 4(b) are the experimental results of the response time attribute, and Figures 4(c) and 4(d) are the experimental results of the throughput attribute. The experimental results in Figure 4 show that λ is one of the decisive factors for RDHE prediction accuracy. As shown in Figures 4(a) and 4(c), as the value of λ increases, the prediction accuracy reaches the maximum value when $\lambda = 0.3$. Meanwhile, as shown in Figures 4(b) and 4(d), the prediction accuracy reaches the maximum value when $\lambda = 0.4$. This shows that a suitable λ value can achieve better prediction accuracy. At the same time, the results in Figure 4 show that when the datasets are different, the optimal value of λ is also different.

5.4.3. The Influence of Top-k for Prediction Accuracy. The parameter k can determine how many similar users or similar services of the selected target are to be used. Here, in order to study the influence of top k-th values on prediction accuracy, we set the matrix density to 0.05 and $\lambda = 0.4$, and set the Top-k value from 1 to 10 and gradually increase at intervals of 1, the results are shown in Figure 5. Among them, Figures 5(a) and 5(b) are the experimental results in the response time attribute, and Figures 5(c) and 5(d) are the experimental results in the throughput attribute. It can be seen from Figures 5(a) and 5(d) that with increasing the top-

Table 1: Default main hyper parameters setting.

Modules	Main hyper parameters	Value
LDA	Number of topics in LDA	20
	Dirichlet parameter	0.1
Node2vec	Length of Node2vec walk	80
	Number of Node2vec walks	10
	Window size of Node2vec walk	10
GCN	Number of layers	2
	Size of layer convolution layer	64
Wide & deep	Number of layers	2
	Dimensions	128
GraphSAGE	Neighborhood deep neighborhood sample sizes	2
		25
NGCF	Embedding propagation layers	3
	Depth of the NGCF	3
	Top-k	20
PinSage	Top-k	2
	Size of hidden dimension size	512
KGAT	Top-k	20
	Tower structure of hidden layer size	512/256/128/64
GraphRec	Size of the hidden layer	128
	Number of three hidden layers	3
NIA-GCN	Size of first layer/second layer	40/2
	Dimension of each layer	64

Table 2: Overall performance comparison of RT on average.

Method	MAE/matrix density						RMSE/matrix density					
	0.10	0.15	0.20	0.25	0.30	Average	0.10	0.15	0.20	0.25	0.30	Average
LDA	0.49	0.46	0.45	0.43	0.42	0.4500	1.48	1.45	1.44	1.43	1.41	1.4420
Node2vec	0.44	0.43	0.41	0.40	0.39	0.4140	1.43	1.41	1.4	1.38	1.37	1.3980
GCN	0.45	0.43	0.43	0.42	0.41	0.4280	1.47	1.45	1.43	1.41	1.39	1.4300
Wide & deep	0.42	0.41	0.39	0.38	0.37	0.3975	1.39	1.36	1.32	1.30	1.28	1.3300
GraphSAGE	0.41	0.4	0.38	0.37	0.36	0.3840	1.38	1.37	1.35	1.34	1.32	1.3520
NGCF	0.4	0.38	0.37	0.38	0.36	0.3780	1.37	1.36	1.35	1.34	1.33	1.3500
PinSage	0.4	0.38	0.36	0.36	0.35	0.3700	1.35	1.34	1.32	1.31	1.30	1.3240
KGAT	0.39	0.38	0.37	0.36	0.35	0.3700	1.31	1.29	1.27	1.26	1.25	1.2760
GraphRec	0.38	0.36	0.35	0.35	0.34	0.3560	1.3	1.27	1.25	1.23	1.22	1.2540
NIA-GCN	0.39	0.36	0.34	0.33	0.32	0.3480	1.29	1.26	1.23	1.22	1.21	1.2420
RDHE	**0.37**	**0.35**	**0.34**	**0.32**	**0.3**	**0.3360**	**1.24**	**1.22**	**1.21**	**1.20**	**1.20**	**1.2140**

Table 3: Overall performance comparison of TP on average.

Method	MAE/matrix density						RMSE/matrix density					
	0.10	0.15	0.20	0.25	0.30	Average	0.10	0.15	0.20	0.25	0.30	Average
LDA	16.29	15.10	14.02	13.97	13.73	14.6220	58.25	57.2	56.56	55.1	54.55	56.3320
Node2vec	14.85	13.76	13.35	13.19	12.82	13.5940	56.85	55.98	54.15	53.29	51.18	54.2900
GCN	15.25	14.30	14.01	13.83	13.55	14.1880	57.93	57.1	55.97	54.35	53.83	55.8360
Wide & deep	14.98	14.21	14.02	13.87	13.85	14.1860	56.09	55.12	54.4	52.97	51.95	54.1060
GraphSAGE	14.76	14.10	13.66	13.11	12.97	13.7200	55.57	54.04	53.52	52.81	53.19	53.8260
NGCF	14.85	14.20	13.86	13.31	12.85	13.8140	55.06	54.02	53.21	51.56	50.94	52.9580
PinSage	14.69	14.15	13.78	13.23	12.95	13.7600	52.87	51.83	50.3	49.78	48.96	50.7480
KGAT	14.76	14.10	13.66	13.11	12.97	13.7200	51.87	51.33	50.25	49.98	48.85	50.4560
GraphRec	14.18	13.53	12.95	12.86	12.75	13.2540	54.23	53.6	52.15	51.42	50.07	52.2940
NIA-GCN	13.58	13.22	13.05	12.99	12.87	13.1420	52.23	51.73	50.05	49.68	48.77	50.4920
RDHE	**12.58**	**12.49**	**12.45**	**12.40**	**12.17**	**12.4180**	**48.26**	**47.41**	**46.16**	**45.32**	**45.01**	**46.4300**

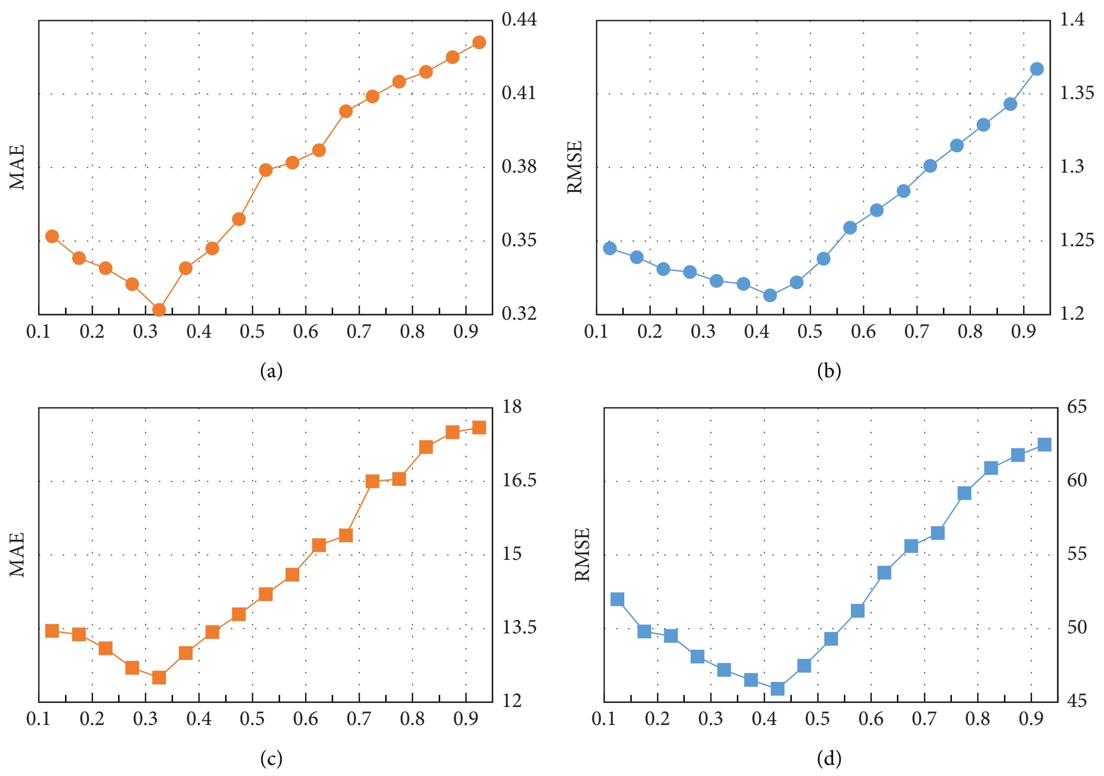

Figure 4: The influence of parameter λ. (a) RT attribute. (b) RT attribute. (c) TP attribute. (d) TP attribute.

k value, the prediction accuracy first increases, then decreases, and reaches the maximum when k is 4. This shows that the number of similar users or similar services is not always beneficial to improving the prediction accuracy, too few or too many similar users and similar services can both reduce the prediction accuracy.

5.4.4. The Influence of Matrix Density. The matrix density represents what proportion of the QoS matrix is used as training data. Figure 6 shows the effects of matrix density on two QoS attributes, response time, and throughput, respectively. Here, we set the density of the matrix from 0.05 to 0.25, increasing in 0.05 intervals, $\lambda = 0.4$, the number of similar users, and the top-k is 4.

We can see from Figure 6 that, as the density of the matrix increases, the service prediction accuracy of the RDHE method gradually improves. This is because the larger the matrix density is, the more users have invoked more services, and the generated data set contains more user information and service information, from which more useful information can be mined to improve prediction accuracy.

5.4.5. The Influence of Hyperparameters. Our method includes three neural networks (wide & Deep, GraphSAGE, and MLP for clustering), among which the default values of the parameters have been depicted in Table 1, more discussion can be referred in [39, 40]. Here, we mainly discuss the key parameters of NN, which are crucial for clustering, including the number of hidden layers, learning rate, and dropout.

(1) Impact of Hidden Layers. In theory, the multi-hidden layer network structure can learn patterns from the original data, and express the data more abstractly through layer-by-layer feature extraction. Therefore, the more layers, the stronger the expressive ability of the data and the stronger the prediction ability. But this improvement is not without an upper limit, and when the number of hidden layers is too high, the accuracy may drop.

To study the number of impact of the hidden layers, we vary the number of hidden layers from 1 to 8, and the dimension of the hidden layer is set to 64. As shown in Figure 7, in both the RT and TP attributes, when the number of hidden layer increases, the value of MAE also increases. When the number of hidden layers is 4, the optimal prediction effect can be achieved. Therefore, we choose the number of hidden layers as 4 in our method.

(2) Impact of Learning Rate. The learning rate is one of the key hyper-parameters for training neural networks. When the learning rate is relatively small, the loss curve converges very slowly, but the amplitude of the loss swing is relatively small because the amplitude of the weight update amplitude is small. Otherwise, when the learning rate is relatively large, the loss curve converges quickly. Therefore, it is very important to choose the learning rate carefully for our method. In order to find the optimal value of the learning rate, we set the learning rate as 0.01, 0.02, 0.03, 0.04, 0.05, and 0.1. Figure 8 shows the performance change of RDHE.

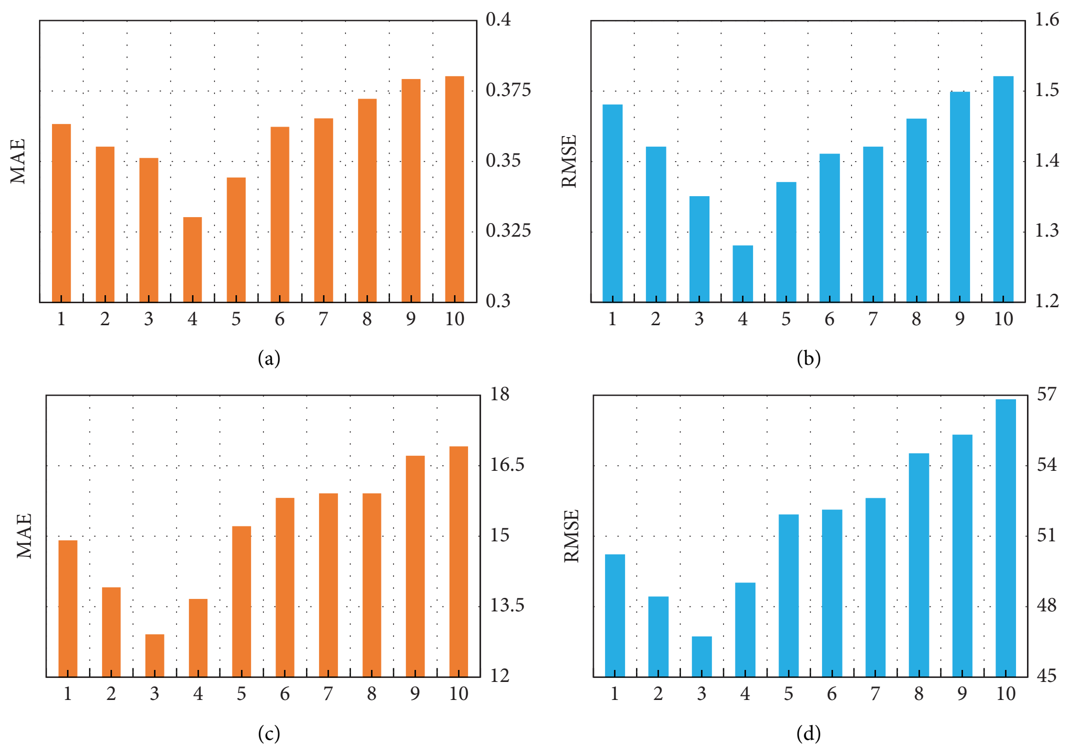

Figure 5: The influence of parameter top-k. (a) Top-k on RT attribute. (b) Top-k on RT attribute. (c) Top-k on TP attribute. (d) Top-k on TP attribute.

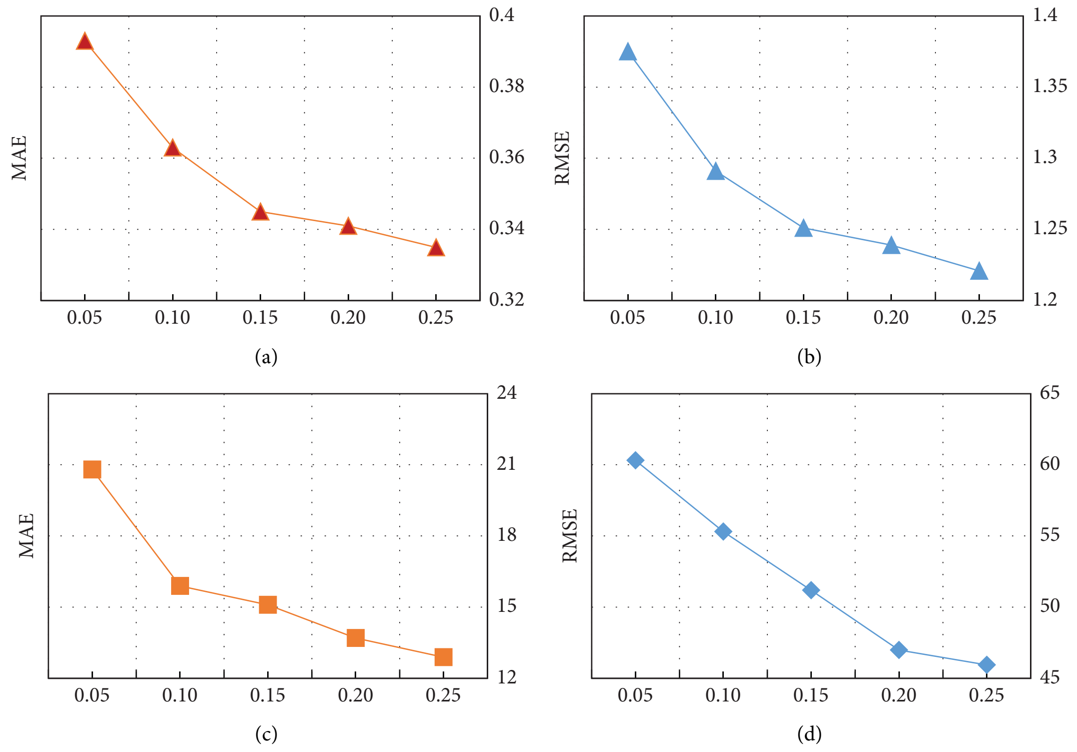

Figure 6: The effect of parameter matrix density. (a) Matrix density on RT. (b) Matrix density on RT. (c) Matrix density on TP. (d) Matrix density on TP.

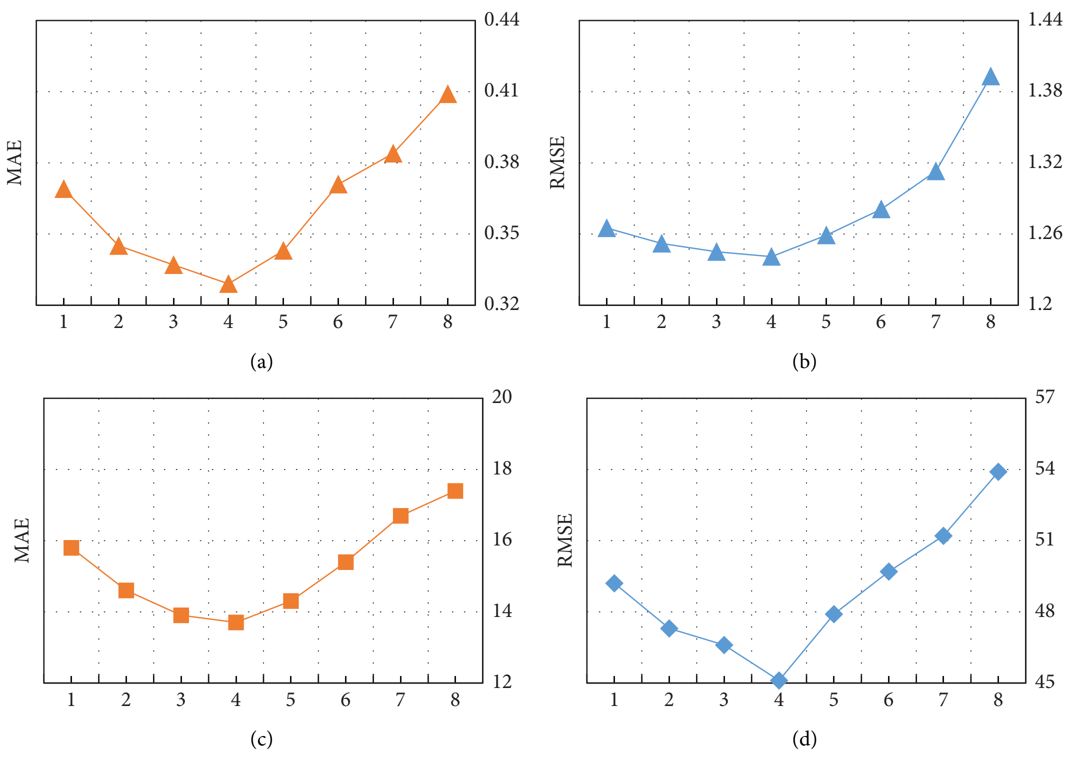

FIGURE 7: Comparison of the hidden layer numbers. (a) Hidden layer number on RT. (b) Hidden layer number on RT. (c) Hidden layer number on TP. (d) Hidden layer number on TP.

From Figures 8(a) and 8(b), we can observe that the learning rate improves the performance when the value range from 0.01 to 0.03 on the RT attribute, and the improvement of the classification effect in the TP attribute does not stop until the learning rate is set to 0.03, this is because a learning rate that is too small can cause the process to get stuck and the method cannot approach the local minima. However, as the value of the learning rate varies from 0.03 to 0.1 on the RT attribute and the TP attribute, the classification performance gradually decreased; the possible cause of this phenomenon is that a learning rate too large will cause the model to converge too quickly to a suboptimal solution. Therefore, in our experiments, the default learning rate is set to 0.03.

(3) Comparative Results of Node and Feature Dropout. Although GCN-based learning models have strong representation ability, they are usually affected by overfitting. The dropout strategy is a common method for solving the overfitting of the neural network, which has the characteristics of simplicity and good effect. Here, we employ node and feature dropout strategies to prevent RDHE from overfitting. The feature dropout strategy randomly removes the output information with a probability of p_n. As a result, only partial information contributes to the refined representations in the l-th propagation layer. We also run a node dropout strategy to randomly block a particular node and discard all its output information with a probability p_f.

Figure 9 shows the effect of the node dropout ratio and the feature dropout on different attributes of the WS-Dream dataset. From Figure 9, we can find that the node dropout strategy gets better accuracy than feature dropout both on (a), (c), and (d). Taking (a) Dropout Ratio on RT as an example, setting the radio as 0.4 will lead to the highest MAE result of 0.329, which is better than that of the feature dropout result of 0.362. One possible reason is that dropping some features of user or service nodes can make the representation more robust; therefore, node dropout is more effective than feature dropout, which means that the node dropout strategy can be a solution to graph neural network overfitting.

5.4.6. Experimental Analysis of Fast Similarity Calculation Method.

In this work, a neural network-based similarity calculation process is designed for downstream similarity calculation and reduced the time complexity. In order to verify the effectiveness of our method, we designed two comparison experiments on TP, (1) FSCM, which had fast similarity calculation component and (2) Non-FSCM, which traversed the embedding one by one directly when calculating $\mathrm{sim}(s_i, s_j)$, the experiment selected five models: LDA, GraphSage, GCN, NGCF, and RDHE for comparison, the unit of Runtime is hours, and the runtime required for the comparison experiment is shown in Table 4.

It can be seen from Table 4 that the addition of the fast similarity calculation component is 4 hours (25.48% less than Non-FSCM) less than the Non-FSCM method on RT and 3.9 hours (24.69% less than Non-FSCM) on TP. It is

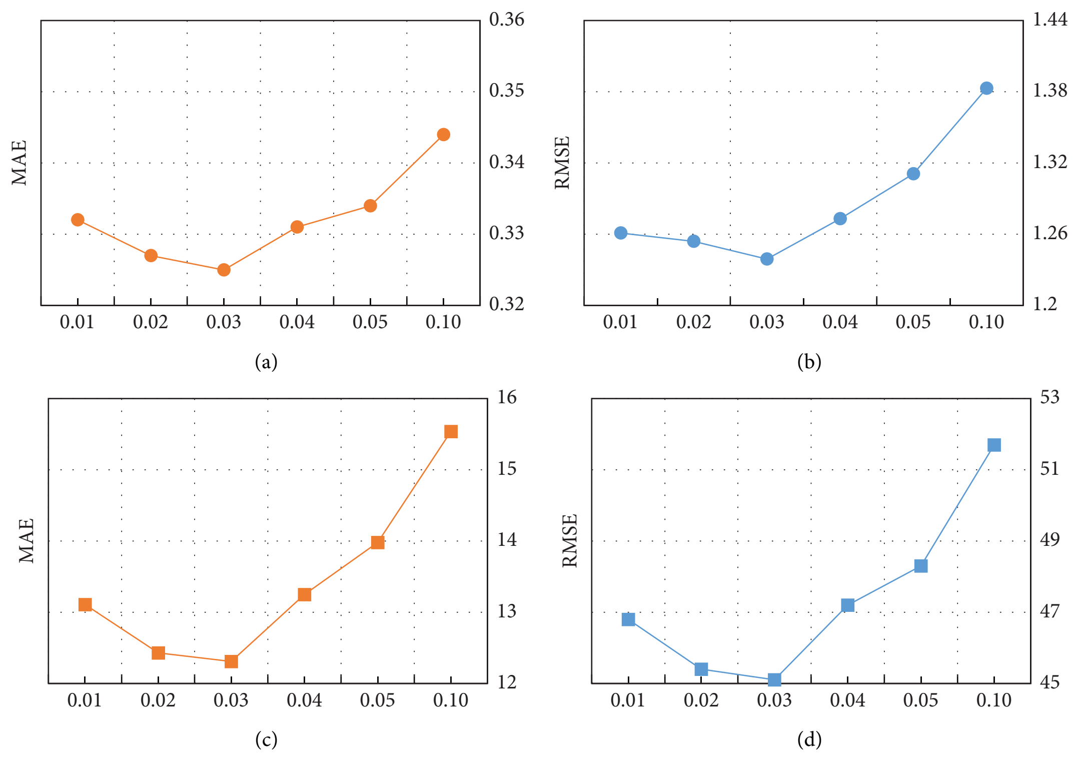

Figure 8: Comparison of the learning rate. (a) Learning rate on RT. (b) Learning rate on RT. (c) Learning rate on TP. (d) Learning rate on TP.

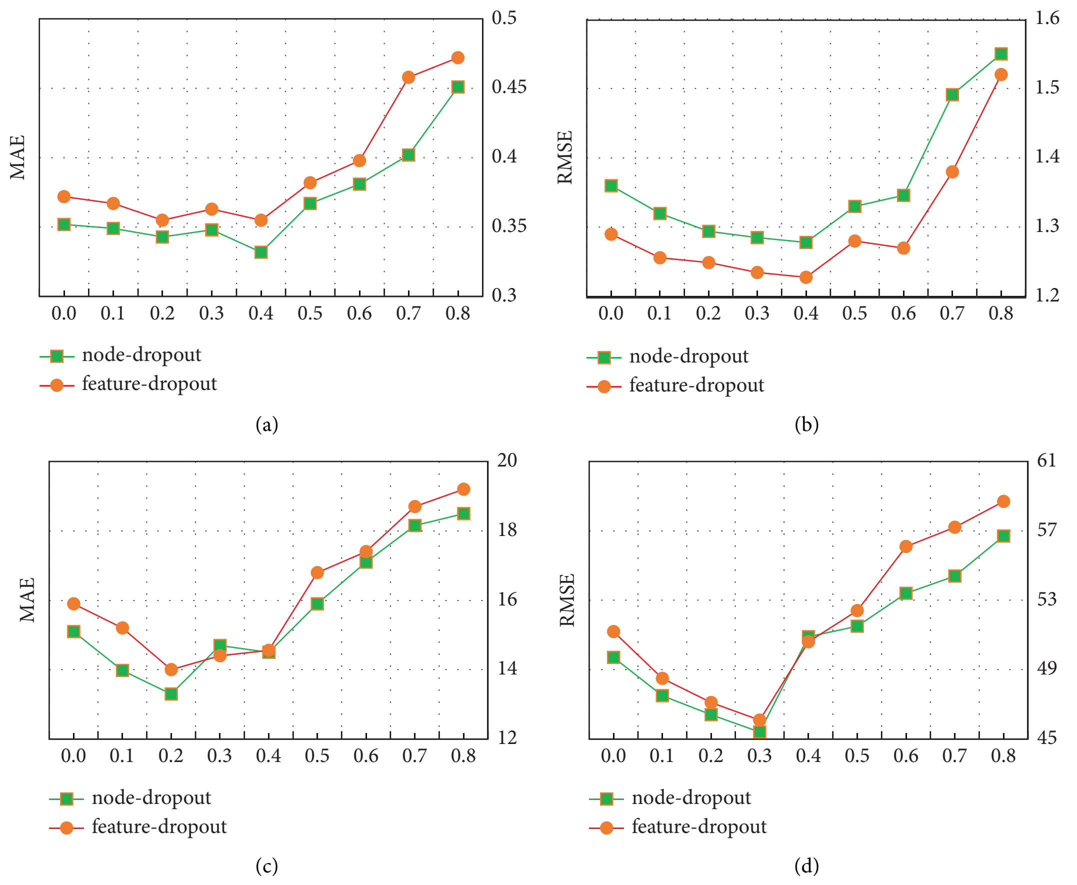

Figure 9: Comparison of dropout ratio. (a) Dropout ratio on RT. (b) Dropout ratio on RT. (c) Dropout ratio on TP. (d) Dropout ratio on TP.

Table 4: Runtime comparison of TP on average (hour).

Models	RT		TP	
	FSCM (h)	Non-FSCM (h)	FSCM (h)	Non-FSCM (h)
LDA	7.5	10.7	7.6	10.9
GraphSage	8.9	14.3	9.1	14.6
GCN	12.1	16.5	12.5	15.9
NGCF	14.7	18.2	14.9	18.4
RDHE	15.2	18.4	15.4	18.6
Average	**11.7**	**15.7**	**11.9**	**15.8**

clear that at the scale of our dataset, the fast similarity calculation method we designed is effective.

6. Conclusions and Future Work

This article proposes a novel community discovery and hierarchical embedding method for web service prediction based on a dual representation and similarity calculation mechanism, which can both utilizes attribute information and structural information to achieve a better prediction effect. We also propose a fast similarity calculation method to improve the algorithm's ability to handle large-scale data, which can avoid the curse of dimensionality of traversing all data samples and reduce the time complexity of similarity calculation. A huge amount of experiments had been carried out on the WS-Dream. The experimental results show that compared to the classic methods (NGCF, PinSage, KGAT, GraphRec, and NIA-GCN), the precision of our RDHE method is up to 11.11%, 9.19%, 9.19%, 5.62%, and 3.45% higher.

Although the experiments have verified the effectiveness of the community discovery method introduced in this article, our method is based on the multi-layer representation of Wide &Deep and GraphSage, the architecture is more complex, and it is based on the convenience of label acquisition, but it is actually a difficult task to obtain labels. According to our analysis and cognition, the next improvement directions are: (1) further research on semi-supervised and even unsupervised methods and (2) further research on end-to-end training, without sub-module or staged training in the learning process to directly optimize the tasks.

Data Availability

The WS-Dream dataset can be obtained at https://github.com/wsdream/wsdream-dataset.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work was supported by Key Projects of Natural Sciences Research in Anhui Universities of China (KJ2016A502), Key research projects of Chaohu University (XLZ-202106), the Key Research and Development Plan of Anhui Province, China (201904a05020091), Anhui Province Teaching Demonstration Course Project (2020SJJXSFK17), and Anhui Province Teaching Research Project (2019jyxm1187).

References

[1] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[2] Y. Syu and C. M. Wang, "QoS time series modelling and forecasting for web services: a comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 18, 2021.

[3] S. Mokarizadeh, P. Küngas, and M. Matskin, "Utilizing web services networks for web service innovation," in *Proceedings of the 2014 IEEE International Conference on Web Services*, pp. 646–653, Anchorage, Alaska, July, 2014.

[4] Y. Cao, J. Liu, M. Shi, B. Cao, X. Zhang, and Y. Wang, "Relationship network augmented web services clustering," in *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS)*, pp. 247–254, Milan, Italy, July, 2019.

[5] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2110–2119, London, UK, August, 2018.

[6] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.

[7] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[9] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, April, 2018.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[11] M. Wang, X. Zheng, Y. Yang, and K. Zhang, "Collaborative filtering with social exposure: a modular approach to social recommendation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[12] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *Proceedings of the World Wide Web Conference*, pp. 417–426, Lyon, France, May, 2019.

[13] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, “Neural graph collaborative filtering,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 165–174, Paris, July, 2019.

[14] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, London, UK, July, 2018.

[15] Y. Xiao, J. Liu, G. Kang, and B. Cao, “LDNM: a general web service classification framework via deep fusion of structured and unstructured features,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3858–3872, 2021.

[16] R. A. H. M. Rupasingha, I. Paik, and B. T. G. S. Kumara, “Specificity-aware ontology generation for improving web service clustering,” *IEICE - Transactions on Info and Systems*, vol. E101.D, no. 8, pp. 2035–2043, 2018.

[17] B. T. Kumara, I. Paik, T. H. A. S. Siriweera, and K. R. Koswatte, “QoS aware service clustering to bootstrap the web service selection,” in *Proceedings of the 2017 IEEE International Conference on Services Computing (SCC)*, pp. 233–240, IEEE, Honolulu, HI, USA, June, 2017.

[18] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Shenzhen, China, July, 2005.

[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[20] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” in *Proceedings of the ICLR*, Austria, April, 2015.

[21] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in *Proceedings of the ICLR*, Vancouver, BC, Canada, August, 2014.

[22] F. Monti, M. M. Bronstein, and X. Bresson, “Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks,” in *Proceedings of the NIPS*, Barcelona, Spain, December, 2017.

[23] M. Zitnik, M. Agrawal, and J. Leskovec, “Modeling polypharmacy side effects with graph convolutional networks,” *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[24] A. Grover and J. Leskovec, “node2vec: scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, San Francisco, CA, USA, August, 2016.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York USA, August, 2014.

[26] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation Learning on Graphs: Methods and Applications,” *IEEE Data Engineering Bulletin*, vol. 40, no. 3, pp. 52–74, 2017.

[27] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, pp. 18–42, 2017.

[28] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, “Heterogeneous graph neural network,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–803, Anchorage AK USA, July, 2019.

[29] Y. Jacob, L. Denoyer, and P. Gallinari, “Learning latent representations of nodes for classifying in heterogeneous social networks,” in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pp. 373–382, Tempe, AZ, USA, February, 2014.

[30] P. Pham and P. Do, “W-MMP2Vec: topic-driven network embedding model for link prediction in content-based heterogeneous information network,” *Intelligent Data Analysis*, vol. 25, no. 3, pp. 711–738, 2021.

[31] Q. Yu, H. Wang, and L. Chen, “Learning sparse functional factors for large-scale service clustering,” in *Proceedings of the 2015 IEEE International Conference on Web Services*, pp. 201–208, IEEE, New York USA, July, 2015.

[32] K. Zeng and I. Paik, “Semantic service clustering with lightweight bert-based service embedding using invocation sequences,” *IEEE Access*, vol. 9, Article ID 54298, 2021.

[33] J. Xu, Z. Zhu, J. Zhao, X. Liu, M. Shan, and J. Guo, “Gemini: a novel and universal heterogeneous graph information fusing framework for online recommendations,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3356–3365, CA USA, August, 2020.

[34] X. Wang, X. He, Y. Cao, M. Liu, and T. S. Chua, “Kgat: knowledge graph attention network for recommendation,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 950–958, Anchorage, AK, USA, August, 2019.

[35] J. Sun, Y. Zhang, W. Guo et al., “Neighbour interaction aware graph convolution networks for recommendation,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1289–1298, Xi’an, China, July, 2020.

[36] J. Yang and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pp. 587–596, Tempe, AZ, USA, July, 2013.

[37] L. Lin, G. Wang, W. Zuo, X. Feng, and L. Zhang, “Cross-domain visual matching via generalized similarity measure and feature learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1089–1102, 2017.

[38] S. Eghbali and L. Tahvildari, “Fast cosine similarity search in binary space with angular multi-index hashing,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 329–342, 2019.

[39] H. T. Cheng, L. Koc, J. Harmsen et al., “Wide & deep learning for recommender systems,” in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7–10, Boston, MA, USA, September, 2016.

[40] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[41] W. L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. J. Hsieh, “Cluster-gcn: an efficient algorithm for training deep and large graph convolutional networks,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge

Discovery & Data Mining, pp. 257–266, Anchorage AK USA, August, 2019.

[42] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[44] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE transactions on services computing*, vol. 7, no. 1, pp. 32–39, 2014.

[45] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.