Knowledge-Based Systems 263 (2023) 110296

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

BGCL: Bi-subgraph network based on graph contrastive learning for cold-start QoS prediction

Jiangyuan Zhu [a], Bing Li [a,*], Jian Wang [a,*], Duantengchuan Li [a], Yongqiang Liu [a], Zhen Zhang [b]

[a] School of Computer Science, Wuhan University, Wuhan, China
[b] School of Information Management, Wuhan University, Wuhan, China

ARTICLE INFO

Article history:
Received 30 September 2022
Received in revised form 17 December 2022
Accepted 9 January 2023
Available online 14 January 2023

Keywords:
Graph contrastive learning
QoS prediction
Web service
Attention mechanism

ABSTRACT

With the advent of Web service technologies, the number of services published on the cloud is increasing rapidly. The quality of service (QoS) becomes a crucial criterion for selecting services from a massive pool of candidates. Collaborative filtering (CF) has become a major way for personalized QoS prediction by leveraging historical interactions between users and services. Due to the increasing number of users and services, CF-based QoS prediction often suffers from data sparsity and cold-start difficulties. Inspired by the advantages of graph contrastive learning in cold-start predictions, we propose BGCL, a bi-subgraph network based on graph contrastive learning to solve the above problems. Firstly, we generate different perspectives of user-neighborhood and service-neighborhood sub-graphs based on sparse user–service bipartite graphs. Next, our model learns user and service embeddings using the graph contrastive learning and graph attention aggregation mechanisms on the generated sub-graphs. Finally, user and service embeddings are fed into a multi-layer perception to predict QoS values. Experimental results show that our model outperforms several existing models in terms of prediction accuracy.

1. Introduction

With the proliferation of Web service technologies, more and more services are being made available in the cloud, making the lives of Internet users much more convenient [1]. How to find and select the most appropriate services from those offering similar functionalities has become a hot topic of research in the services computing society in response to the exponential growth of services. Non-functional properties, also known as quality of services (QoS), have become primary considerations in the decision-making of service selection.

QoS refers to the non-functional characteristics of Web services [2,3], including response time, throughput, reliability, cost, and so on. In general, QoS values can be monitored on both the provider and user sides. Some QoS values, such as cost, can be set by service providers, but the majority of QoS values are determined by a variety of factors, e.g., the varying network environment and geographic locations of service users. Various users may observe different QoS values when invoking the same service. Consequently, user-side QoS evaluations, which are more pertinent to individualized user experiences on Web services, are receiving more consideration. In services computing, it is crucial to precisely predict the unknown QoS values of Web services for various users, which can considerably enhance the user experiences on services [4–8].

Fig. 1 illustrates how users may utilize Web services, which can be modeled as an interaction matrix between users and services. QoS prediction seeks to forecast the QoS values of each service invoked by a target user using a limited amount of data. The first challenge for this task is the data sparsity [9], which is caused by the fact that the number of services and users is often enormous, yet each user may have only invoked a small number of services. In the case of sparse data, it is hard to achieve satisfactory recommendation results, especially when utilizing conventional collaborative filtering approaches. The cold-start problem is the second challenge. Due to the lack of historical invocation records, it is difficult to extract their high-dimensional features for prediction purposes when a new user or a new service is introduced [10,11].

Earlier research in this field relied mostly on memory-based collaborative filtering techniques, such as user-based and item-based collaborative methods, for QoS prediction [12,13]. Memory-based collaborative filtering predicts missing QoS values by finding users with similar characteristics to the target users based

* Corresponding authors.
E-mail addresses: bingli@whu.edu.cn (B. Li), jianwang@whu.edu.cn (J. Wang).

https://doi.org/10.1016/j.knosys.2023.110296
0950-7051/

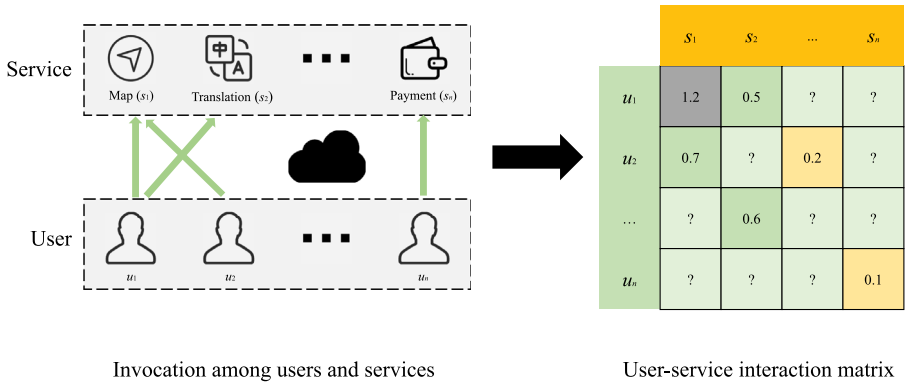

Fig. 1. Illustration of how users invoke Web services.

on historical data. Later on, some researchers employ techniques like matrix factorization to learn the representations of users and services [14]. With the development of the neural network, deep neural networks such as the multilayer perceptron (MLP) and graph neural networks have also been introduced for QoS prediction. Nevertheless, these existing methods are incapable of mining well-behaved embedding representations, resulting in poor performance in low-density and cold-start situations.

Considering that user–service interactions are typically graph-structures data, we introduce graph networks into QoS prediction to learn the representation of users and services [15–17]. Different from existing methods, we take into account historical invocation data and geographic location information to construct interaction graphs, and then mine well-preformed features based on the graph attention network.

Faced with the sparse interaction data in real-world scenarios, we introduce a data enhancement method named edge dropout to construct sub-graphs from different perspectives based on the existing data, so as to learn additional features from the limited observed data using graph contrastive learning.

To sum up, the primary contributions of this paper are listed as follows:

1. We propose BGCL, a bi-subgraph network based on graph contrastive learning framework, which constructs an adjacency graph by fusing location similarity and invocation similarity and then integrates geographical location and graph attention network to mine more latent fine-grained features, hence improving the accuracy of QoS prediction.
2. Because the observable interaction data is extremely sparse, a data augmentation mechanism called edge dropout is employed to enrich the dataset. Graph contrastive learning is also utilized to extract implicit data from the graph structure.
3. Extensive experiments conducted on WS-DREAM, a large-scale real-world dataset, indicate that our model is more accurate in cold-start QoS prediction than several conventional approaches.

The remainder of this paper is organized as follows. Section 2 reviews the work related to QoS prediction. The proposed model is presented in Section 3. We exhibit the experimental results in Section 4. Finally, we conclude the paper with future research directions in Section 5.

2. Related work

2.1. QoS prediction

QoS prediction has been investigated in depth in the past decade. The majority of advancements in this field are based on collaborative filtering [18–23], which can be roughly classified into two categories: memory-based QoS prediction approaches and model-based QoS prediction approaches.

2.1.1. Memory-based QoS prediction approaches

Memory-based QoS prediction computes the similarity between users or services based on historical QoS data to determine their nearest neighbors. It then forecasts QoS values based on the target user's nearest neighbors with the highest degree of similarity. The memory-based collaborative filtering algorithm is further subdivided based on the neighbors into user-based, item-based, and hybrid collaborative filtering algorithms [24,25]. Nevertheless, these memory-based methods are ineffective when the user–service matrix becomes larger and sparser as the number of users and services increases. The feature of users and services cannot be completely utilized with the existing data.

2.1.2. Model-based QoS prediction approaches

The model-based algorithm represents the recommendation task as a representation learning problem. Methods such as matrix decomposition, clustering, hidden semantic model, Bayesian and neural networks, etc., are utilized frequently. Non-negative matrix factorization (NMF) utilizes non-negative matrix factorization to predict QoS values [26]. After aggregating users and services based on geographic location information, hierarchical matrix factorization (HMF) utilizes local factorization and global factorization to forecast QoS [27]. Network-aware matrix factorization uses a network map to factorize in order to forecast QoS [28]. Location-based matrix factorization technique via preference propagation (LMF-PP) employs invocation and neighborhood information in the process of preference propagation to predict the missing QoS values [29]. In conclusion, rather than depending on the selection of similar neighbors, the prediction accuracy of the model-based CF technique is often superior to that of the memory-based CF method in the presence of sparse data. However, in a cold-start scenario, it cannot learn the latent characteristics of users and services, making it difficult to generate correct predictions.

There have been a great number of works applying graph neural networks to recommender systems due to their capability in learning from graph data [30]. Graph convolutional matrix completion provides a graph auto-encoder architecture that produces latent features of users and items via a form of differentiable message passing on the user–item graph in order to tackle the rating prediction problem in recommender systems from the perspective of link prediction [31]. Neural graph collaborative filtering is proposed to explicitly encode user/item signals in the form of high-order connectivities by graph propagation item recommendations [32].

A few further studies introduced GNN to mine well-behaved representations for better QoS prediction. Li et al., for instance, proposed a graph matrix factorization approach, which combines GNN and collaborative filtering to estimate missing QoS values in the data matrix [33]. Ding et al. proposed MGCCF-DFM, a multi-component graph convolutional collaborative filtering and deep factorization machine model. MGCCF-DFM uses GCN to extract user preference from the service environment [34]. Chang et al. proposed a graph-based matrix factorization approach to consolidate various multi-source information, and further dig out potential relationships in the graph model to obtain QoS values [35].

The aforementioned studies contribute significantly to the advancement of QoS prediction techniques. For example, average values from the past are used for prediction because they are a good representation of the characteristics of users and services that tend to remain constant over time. Another example is the consideration of geographical location similarity is based on the close correlation between network conditions and geographic locations. However, few of them comprehensively consider historical invocations and geographic locations. Moreover, it is difficult for existing methods to mine well-performing representations with good performance from sparse data due to limitations imposed by the model design.

The model proposed in this paper has two key characteristics. First, a data augmentation mechanism called edge dropout is introduced to enrich the dataset for mining more implicit representations from the limited data. Second, high-dimensional representations are extracted from graph structures constructed with comprehensive consideration of historical invocation and geographic location similarity using a graph attention network.

2.2. Contrastive learning

Contrastive learning (CL) has recently gained popularity as a technique for unsupervised graph representation learning. CL has been implemented extensively in computer vision, natural language processing, graph data mining, and recommender systems [36,37]. Self-Supervised learning for sequential recommendation (S3-Rec) was proposed to utilize the intrinsic data correlation to derive self-supervision signals and enhance the data representations via pre-training methods for improving sequential recommendation [38]. Yao et al. proposed a framework to tackle the label sparsity problem by learning better latent relationships of item features based on multi-task self-supervised learning [39]. Self-supervised graph learning for recommendation (SGL) explored self-supervised learning on the user–item graph, so as to improve the accuracy and robustness of GCNs for recommendation.

These methods are typically applied in computer vision or tasks with specific graph structures. Existing methods are unable to model the QoS prediction task for feature learning due to the characteristics of sparse data and lack of evident graph structure information. Therefore, we propose a bi-subgraph graph contrastive learning network for cold-start QoS prediction, which exploits the implicit graph structure based on location data and previous invocation records. The dataset is then enhanced using the edge dropout approach for mining well-performed implicit information.

3. Methodology

Suppose that there are n_u users $\mathcal{U} = \{u_1, u_2, \ldots, u_{n_u}\}$, n_s Web services $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_s}\}$ and n_r QoS records $\mathcal{E} = \{e_1, e_2, \ldots, e_{n_r}\}$. The element of $\mathcal{E}$ edge $e = (u, s, r)$ indicates that there is an observed QoS value r from user u to service s. The user–service QoS matrix can be modeled as a user–service bipartite graph $\mathcal{G} = \{\mathcal{U}, \mathcal{S}, \mathcal{E}\}$. In reality, n_u and n_s are very large, and only limited interaction data can be observed, making it difficult to obtain precise predictions.

In this paper, we propose a model named bi-subgraph network based on graph contrastive learning (BGCL) to alleviate the above problem. Fig. 2 depicts the architecture of our model, which consists of three parts: Bi-subgraph generation, graph contrastive learning, and QoS prediction.

- Bi-subgraph generation: The strategy of edge dropout is applied to the original dataset for data augmentation, yielding user-neighborhood sub-graphs and service-neighborhood sub-graphs.
- Graph contrastive learning is introduced to solve the problem of data sparseness, aiming to mine well-preformed embedding representations for users and services from the limited data.
- QoS prediction: The QoS prediction part integrates the user embeddings and the service embeddings through an MLP to achieve precise QoS prediction.

In the following, we present the principles and implementation steps for each part of the BGCL model.

3.1. Bi-subgraph generation

The observed interactions are extremely sparse compared to the whole interaction space, making it inadequate to learn high-quality representations. Because it is helpful for representation learning to increase the amount of trained data using data augmentation, we introduce a strategy named edge dropout to generate multiple representation views, which will in turn learn more high-dimensional information. It removes edges from the graph with a dropout ratio p. In this paper, we apply the edge drop strategy to construct two sub-graphs as follows:

$$s_1(\mathcal{G}) = (\mathcal{U}, \mathcal{S}, \mathbf{M}_1 \odot \mathcal{E}, p), \quad s_2(\mathcal{G}) = (\mathcal{U}, \mathcal{S}, \mathbf{M}_2 \odot \mathcal{E}, p), \tag{1}$$

where $\mathcal{G}$ denotes the user–service interaction graph, $\mathcal{U}$ is the user set, $\mathcal{S}$ is the service set and $\mathbf{M_1}, \mathbf{M_2} \in \{0, 1\}^{|\varepsilon|}$ are two masking vectors on the edge set ε, and p is the dropout ratio. Only partial connections within the neighborhood contribute to the node representations. The integration of these two sub-graphs aims to capture the instructive patterns of the local structures of a node and further endow the representations with more robustness against noisy interactions.

Considering the fact that user nodes and service nodes belong to different feature spaces, the latent features of the user may include the network environment of the user, while the latent features of the service contain the function of the service. Clearly, it is unreasonable to directly aggregate node features based on the user–service bipartite graph. As a result, we construct a user adjacency sub-graph and a service-adjacency sub-graph based on user–service interaction data for subsequent aggregation training, which facilitates the extraction of exact high-dimensional features.

Inspired by conventional collaborative filtering, invocation similarity and location information are regarded as two crucial elements for creating adjacency sub-graphs. On the one hand, a higher invocation similarity indicates that two users or services may share comparable network conditions. Furthermore, two nearby users or services should have similar high-dimensional location embedding information.

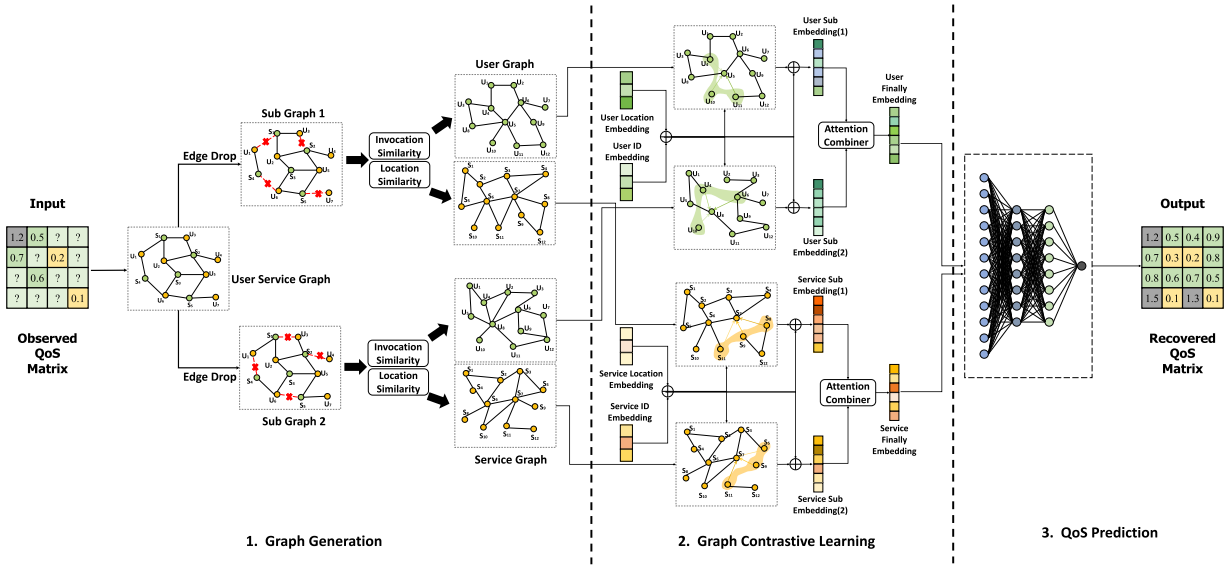

Fig. 2. The overview of BGCL framework.

3.1.1. Invocation similarity

Invocation similarity is constructive for mining implicit information from historical interaction data. Pearson correlation coefficient (PCC) has been adopted for similarity calculation in numerous recommendation tasks due to its high accuracy. In this paper, we calculate the invocation similarity using PCC. The degree of invocation similarity between two users is defined as:

$$S_{inv}(u_i, u_j) = \frac{\sum_{s \in I_i^U \cap I_j^U}(q_{i,s} - \bar{q}_i)(q_{j,s} - \bar{q}_j)}{\sqrt{\sum_{s \in I_i^U \cap I_j^U}(q_{i,s} - \bar{q}_i)^2}\sqrt{\sum_{s \in I_i^U \cap I_j^U}(q_{j,s} - \bar{q}_j)^2}}, \tag{2}$$

where $s \in I_i^U \cap I_j^U$ is a set of Web services that are commonly called by both users i and j. $q_{i,s}$ denotes the historical QoS value of Web service s invoked by user i. $\bar{q}_i$ is the average historical QoS value of user i. The value of $S_{inv}(u_i, u_j)$ is in $(-1, 1)$.

Similarly, the degree of invocation similarity between two services is defined as:

$$S_{inv}(s_m, s_n) = \frac{\sum_{u \in I_m^S \cap I_n^S}(q_{u,m} - \bar{q}_m)(q_{u,n} - \bar{q}_n)}{\sqrt{\sum_{u \in I_m^S \cap I_n^S}(q_{u,m} - \bar{q}_m)^2}\sqrt{\sum_{u \in I_m^S \cap I_n^S}(q_{u,n} - \bar{q}_n)^2}}, \tag{3}$$

where $u \in I_m^S \cap I_n^S$ is a set of users who invoked services m and n. $q_{u,m}$ denotes the historical QoS value of Web service m invoked by user u and $\bar{q}_m$ is the average historical QoS value of service m.

3.1.2. Location similarity

Location similarity can be computed even in the absence of interaction history, thereby alleviating the cold-start problem. Considering that both the user and the service had latitude and longitude coordinates, which could be utilized to determine the distance between two locations. For the subsequent computation of geographic similarity, we introduce the haversine formula to compute the distance between two locations based on latitude and longitude. Haversine is defined as follows.

$$D(i,j) = 2r \arcsin \cdot \left(\sqrt{\sin^2\left(\frac{\phi_i - \phi_j}{2}\right) + \cos(\phi_j)\cos(\phi_i)\sin^2\left(\frac{\varphi_j - \varphi_i}{2}\right)}\right), \tag{4}$$

where $D(i,j)$ is the distance between places i and j, ϕ_i is the longitude of place i, φ_i is the latitude of i, ϕ_j is the longitude of place j, and φ_j is the latitude of j.

Once the user-to-user distance matrix has been generated, the associated user location similarity can be determined, as shown below.

$$S_{loc}(u_m, u_n) = \frac{e^{-D(u_i,u_j)}}{\sqrt{\delta_u}}, \tag{5}$$

where $D(u_i, u_j)$ indicates the distance between u_i and u_j, and δ_u is the variance of the user distance matrix.

Similarly, the location similarity of services can be calculated as:

$$S_{loc}(s_m, s_n) = \frac{e^{-D(s_m,s_n)}}{\sqrt{\delta_s}}, \tag{6}$$

where $D(s_m, s_n)$ indicates the distance between s_m and s_n, and δ_s is the variance of the service distance matrix.

After obtaining the two similarity rankings, the two similarities are combined to produce the final node similarity ranking. Considering that the invocation similarity is affected by geographical location, a weight mechanism is implemented to balance the two similarities.

$$S = \alpha S_{inv} + (1 - \alpha) S_{loc}, \tag{7}$$

where α is the weight setting of invocation similarity.

Once the final similarity ranking is determined, we take the *top-k* nodes for each node to build the corresponding adjacency graph, thereby obtaining the user adjacency, A_1^U, A_2^U, and the service adjacency, A_1^S, A_2^S, from $S_1(\mathcal{G})$ and $S_2(\mathcal{G})$, respectively, as shown in Fig. 2.

3.2. Graph contrastive learning

To extract the high-dimensional characteristics from the constructed user adjacency sub-graph and service adjacency sub-graph, we employ the graph attention network (GAT) for neighbor aggregation in each case.

3.2.1. Embedding layer

The features we selected for initial embedding include the user ID, the geolocation information of the user, the service ID, and the geolocation information of the service. Taking the user in sub-graph m as an example, the user ID is embedded into high-dimensional space $\mathbf{U}_{m_i} = [\mathbf{u}_{m_i}^1, \mathbf{u}_{m_i}^2, \ldots, \mathbf{u}_{m_i}^{n_u}] \in \mathbb{R}^K$, then the

user's geolocation information is embedded into another high-dimensional space as well $\mathbf{U}_{m_l} = [\mathbf{u}^1_{m_l}, \mathbf{u}^2_{m_l}, \ldots, \mathbf{u}^{n_u}_{m_l}] \in \mathbb{R}^K$. These two features are combined through concatenation operations to represent the users' whole high-dimensional information $\mathbf{U}_m = [\mathbf{U}_{m_i} \parallel \mathbf{U}_{m_l}] = [\mathbf{u}^1_m, \mathbf{u}^2_m, \ldots, \mathbf{u}^{n_u}_m] \in \mathbb{R}^{2K}$. Similarly, for services, the service ID is embedded into high-dimensional space $\mathbf{S}_{m_i} = [\mathbf{s}^1_{m_i}, \mathbf{s}^2_{m_i}, \ldots, \mathbf{s}^{n_s}_{m_i}] \in \mathbb{R}^K$ and the geolocation information of service will be embedded to another space $\mathbf{s}_{m_l} = [\mathbf{s}^1_{m_l}, \mathbf{s}^2_{m_l}, \ldots, \mathbf{s}^{n_s}_{m_l}] \in \mathbb{R}^K$. They are combined to obtain the embedding representation for service $\mathbf{S}_m = [\mathbf{S}_{m_i} \parallel \mathbf{S}_{m_l}] = [\mathbf{s}^1_m, \mathbf{s}^2_m, \ldots, \mathbf{s}^{n_s}_m] \in \mathbb{R}^{2K}$. It is worth noting here that the identification information and geolocation information of user and service are embedded in the K-dimensional space. The embeddings of users and services are randomly initialized with the Gaussian distribution and further optimized with model training.

3.2.2. Graph attention layer

To learn higher-dimensional node features from the graph structure for high-precision QoS prediction, graph attention network is introduced to the user adjacency graph and the service adjacency graph for representation learning.

Unlike GCN, which aggregates neighboring features with fixed weights determined by the degrees of the corresponding nodes, GAT uses learnable self-attention-based weights to discriminate the relative relevance of surrounding features. The feature update processes in the GAT layer are conducted in two steps: weight calculation and feature aggregation.

Taking the user sub-graph m as an example. For each node in the graph, the final features can be obtained from the neighbor aggregation.

$$\alpha^{ij}_m = att(\mathbf{u}^i_m, \mathbf{u}^j_m) = \frac{\exp\left(\sigma\left(\mathbf{a}^{\mathrm{T}}_m \cdot \left[\mathbf{u}^i_m \parallel \mathbf{u}^j_m\right]\right)\right)}{\sum_{j\in\mathcal{P}^i_m} \exp\left(\sigma\left(\mathbf{a}^{\mathrm{T}}_m \cdot \left[\mathbf{u}^i_m \parallel \mathbf{u}^j_m\right]\right)\right)}, \tag{8}$$

$$\mathbf{z}^u_m = \sigma\left(\sum_{j\in\mathcal{P}^i_m} \alpha^{ij}_m \cdot \mathbf{u}^j_m\right), \tag{9}$$

where α^{ij}_m indicates the importance weight of the user j's feature to user i in sub-graph m, $\mathbf{u}^i_m$ is the embedding representation of user i in sub-graph m, $\parallel$ is the concatenation operation, $\mathbf{a}^{\mathrm{T}}_m$ is the transform matrix for sub-graph m. σ is the softmax function. $\mathbf{z}^u_m$ is the final embedding representation of user in sub-graph m. $\mathcal{P}^i_m$ denotes the neighborhood of node i in sub-graph m.

3.2.3. Contrastive learning

After establishing the various sub-graphs, we treat the same node in the sub-graphs as positive pairs and the different nodes as negative pairs. We assume that some high-dimensional features of the node are fixed, such as the location information of the user and the network environment, the functions provided by the service and the server configuration, and so on. Because the high-level features are also fixed, they should be as close to the different views as possible to indicate the high-level abstract features of users and services. This means that the embedding of the positive pairs should be as close as possible, while the features of the negative pairs should be as far away as possible. We follow SIMCLR to introduce a contrastive learning loss to maximize the agreement of positive pairs and minimize that of negative pairs [40]. The contrastive learning loss is defined as follows.

$$L^{user}_{ssl} = \sum -\log \frac{exp(s(z^u_1, z^u_2)/\tau)}{\sum exp(s(z^u_1, z^u_2)/\tau)}, \tag{10}$$

where s(·) measures the similarity between two vectors using the cosine similarity function, τ is the hyper-parameter, known as the temperature in softmax. Analogously, we obtain the contrastive loss of the service side by combining these two losses, we get the objective function of the contrastive learning task as $L_{ssl} = L^{user}_{ssl} + L^{service}_{ssl}$.

3.2.4. Combiner

It is well recognized that the QoS value is usually determined by many factors, which can be learned from different views of the node. Therefore, different views of the node should have different contributions to learning the final embedding of users, motivating us to introduce the attention mechanism to automatically discover the importance of different views.

$$\left(\beta^u_1, \beta^u_2\right) = att_{com}\left(\mathbf{z}^u_1, \mathbf{z}^u_2\right), \tag{11}$$

$$\mathbf{z}_u = \sum_{m=1}^{2} \beta^u_m \cdot \mathbf{z}^u_m, \tag{12}$$

where β^u_1 and β^u_2 are the importance weights of A^U_1 and A^U_2, respectively. $\mathbf{z}_u$ is the final embedding of user.

Similarly, the final service embedding can be obtained as:

$$\left(\gamma^s_1, \gamma^s_2\right) = att_{com}\left(\mathbf{z}^s_1, \mathbf{z}^s_2\right), \tag{13}$$

$$\mathbf{z}_s = \sum_{m=1}^{2} \gamma^s_m \cdot \mathbf{z}^s_m, \tag{14}$$

where γ^s_1 and γ^s_2 are the importance weights of A^S_1 and A^S_2, respectively. $\mathbf{z}_s$ is the final embedding of service.

3.3. QoS prediction

Once the representations of user $\mathbf{Z}_u$ and service $\mathbf{Z}_s$ from the user part and service part are obtained, we concatenate them using an MLP to predict the QoS value r from user u to service s.

$$\mathbf{g}_1 = \sigma\left(\mathbf{W}_1 \cdot \mathbf{Z}_u + \mathbf{b}_1\right), \tag{15}$$

$$\mathbf{g}_2 = \sigma\left(\mathbf{W}_2 \cdot \mathbf{Z}_s + \mathbf{b}_2\right), \tag{16}$$

$$\mathbf{r}_{u,s} = \sigma\left(\mathbf{W}_3 \cdot [\mathbf{g}_1 \parallel \mathbf{g}_2] + \mathbf{b}_3\right). \tag{17}$$

3.4. Complexity analysis

The pseudo code of BGCL is shown in Algorithm 1. Assume that N_u, N_s, and N represent the number of users, services, and neighbors, respectively. The proposed model can be roughly divided into three parts: graph generation, graph contrastive learning, and QoS prediction. First, in the graph generation phase, BGCL calculates the similarity of users and services. The time complexity of this part is $O(N^2_u + N^2_s)$. Next, BGCL aggregates the neighbor information in the generated subgraphs based on the graph attention network. The time complexity is $O(2 * N * N^2_u + 2 * N * N^2_s)$. Finally, the learned embedded representation is fed into the MLP for QoS prediction. The time complexity is $O(N_u + N_s)$. To sum up, the time complexity of the whole algorithm is $O(N^2_u + N^2_s)$.

4. Experiments

4.1. Dataset

To evaluate the performance of our model, a series of experiments are conducted on WS-DREAM [41], a large-scale real-world QoS dataset. The dataset consists of 1,974,675 QoS records (including response time and throughput) obtained from 339 users with 5825 Web services distributed in 73 countries. In addition

Algorithm 1 Framework of BGCL

Input:
the user-service interaction graph $\mathcal{G}$;
the information of users $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N_u}]$;
the information of services $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{u}_{N_s}]$;
the weight of similarity α ;
the rate of dropout P ;
the number of neighbors N.

// PART 1: Graph Generation
Drop the egdes base on P to generate sub-graphs $\mathcal{G}_1$ and $\mathcal{G}_2$
for $i = 1$ to 2 **do**
  // generate sub-graphs for $\mathcal{G}_i$
  for $j = 1$ to N_u **do**
    // Calculate the invocation similarity for $\mathbf{u}_j$;
    InvSimi($\mathbf{u}^i$)
    // Calculate the location similarity for $\mathbf{u}_j$;
    LocSimi($\mathbf{u}^i$)
  end for
  for $k = 1$ to N_s **do**
    // Calculate the invocation similarity for $\mathbf{s}_k$;
    InvSimi($\mathbf{s}^j$)
    // Calculate the location similarity for $\mathbf{s}_k$;
    LocSimi($\mathbf{s}^j$)
  end for
  Take the top-N nodes for each node to build the corresponding adjacency graph A_i^U and A_i^S for $\mathcal{G}_i$
end for
// PART 2: Graph Contrastive Learning
// Graph attention layer
GAT($A_1^U, A_2^U, A_1^S, A_2^S$)
// Contrastive learning
SSL($A_1^U, A_2^U, A_1^S, A_2^S$)
// PART 3: QoS Prediction
MLP($\mathbf{Z}_U$,$\mathbf{Z}_S$)

to interaction information between users and services, the dataset also includes basic user and service attributes. The user attributes consist of [ID, IP Address, Country, IP NO., AS (Autonomous System), Latitude, Longitude], whereas the service attributes include [ID, WSDL Address, Service Provider, IP Address, Country, IP NO., AS, Latitude, Longitude]. Their real location attributes are employed in our proposed approach. We make our code available for further study.[1]

4.2. Evaluation metrics

We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to measure the performance of the proposed model and other baselines. Lower scores for these two indicators suggest a more accurate prediction.

- Mean Absolute Error: MAE is the average of the absolute values of the deviations of all individual observations from the arithmetic mean.

$$\text{MAE} = \frac{\sum_{i=1}^{N} |\mathbf{Q}_{u,s} - \hat{\mathbf{Q}}_{u,s}|}{N}. \tag{18}$$

- Root Mean Square Error: RMSE indicates the deviation between the observed value and the truth.

$$\text{RMSE} = \frac{\sum_{i=1}^{N} (\mathbf{Q}_{u,s} - \hat{\mathbf{Q}}_{u,s})^2}{N}, \tag{19}$$

where $\mathbf{Q}_{u,s}$ is the actual QoS value of service s observed by user u, $\hat{\mathbf{Q}}_{u,s}$ is the predictive QoS value of service s observed by user u, and N denotes the total number of QoS values.

4.3. Baselines

To evaluate the performance of the proposed approach, we selected ten baselines for comparison, including the traditional algorithms and the latest research.

In general, the selected baselines can be categorized into two groups: memory-based and model-based techniques. UMEAN, IMEAN, UPCC, IPCC and UIPCC are memory-based prediction methods. These methods essentially utilize user and item-related history for the recommendation. The remaining five baselines are model-based methods. Among them, NMF and HMF are the most prominent methods based on matrix factorization. LMF-PP takes into account the invocation similarity. LDCF utilizes geographic location information. MGCF-DFM integrates graph networks into the QoS prediction process. To the best of our knowledge, MGCF-DFM is the latest method for QoS prediction, and LMF-PP is the latest research on addressing the cold-start problem.

- UMEAN [12]: This approach predicts missing values by averaging the available QoS values based on the target user
- IMEAN [13]: This approach predicts missing values by averaging the available QoS values base,d on the target Web service.
- UPCC [12]: This approach is one of the memory-based collaborative filtering, which calculates the user's similar neighbors by PCC, then predicts missing QoS by considering the historical information of the user and its neighbors.
- IPCC [13]: This approach is similar to UPCC, which calculates the services' similar neighbors by PCC, then predicts missing QoS by considering the historical information of the user and its neighbors.
- UIPCC [13]: This approach is a hybrid approach that linearly combines UPCC and IPCC, which computes similar users and similar services by PCC and combines them to recommend services to target users.
- NMF [26]: This is a model-based collaborative filtering approach, applying a constraint that all factorized values must be nonnegative to predict QoS values.
- HMF [27]: This is a hierarchical MF approach, which clusters users and services with the location information, then combines the prediction of local matrix factorization and global matrix factorization.
- LMF-PP [29]: This approach employs invocation and neighborhood information in the process of preference propagation to predict the missing QoS values.
- LDCF [42]: This approach learns the high-dimensional and nonlinear relationships between users and services through an MLP and incorporates the similarity module to correct the predictive values.
- MGCF-DFM [35]: This approach extracts the high-dimensional via combining multi-component graph convolutional collaborative filtering and deep factorization machine.

4.4. Comparison of prediction accuracy for high-sparsity

To evaluate the performance of our model for high sparsity, we varied the density of the original dataset by 0.5%, 1%, 2%, 3%, and 4%, in accordance with the state-of-the-art work on high-sparsity QoS prediction [29]. For example, a density of 0.5 means that 0.5% of the original 1,974,675 interactions are selected for training, and the remaining 99.5% of the data are used for prediction.

[1] https://github.com/zjy5755202/BGCL.

Table 1
Performance comparison on response time prediction in the high-sparsity situation.

addressesfMethods	Density = 0.5%(D1)		Density = 1%(D2)		Density = 2%(D3)		Density = 3%(D4)		Density = 4%(D5)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
UMEAN [12]	1.8847	0.9015	1.8721	0.8992	1.8638	0.8783	1.8607	0.8754	1.8596	0.8774
IMEAN [13]	1.9094	0.8019	1.7588	0.7632	1.6424	0.7345	1.6010	0.7187	1.5787	0.7113
UPCC [12]	1.8855	0.9130	1.8828	0.8999	1.8846	0.8946	1.8680	0.8655	1.7920	0.8076
IPCC [13]	2.0062	0.8381	1.8253	0.7875	1.6651	0.7411	1.6092	0.7186	1.5784	0.7063
UIPCC [13]	1.8696	0.8119	1.7549	0.7903	1.6661	0.7600	1.6067	0.7315	1.5631	0.7110
NMF [26]	2.1218	0.8920	2.0740	0.8576	1.8269	0.7099	1.6555	0.6219	1.5482	0.5749
HMF [27]	2.0102	0.7560	1.9585	0.7348	1.8764	0.6942	1.7432	0.6529	1.5947	0.5987
LMF-PP [29]	1.8668	0.7256	1.7236	0.6684	1.5554	0.6021	1.4767	0.5733	1.4339	0.5577
LDCF [42]	1.8708	0.7042	1.7111	0.7042	1.4703	0.4917	1.4069	0.4727	1.3513	0.4330
MGCF-DFM [35]	1.7693	0.6743	1.6144	0.6028	1.4618	0.5297	1.4077	0.5325	1.3792	0.5134
BGCL	**1.6813**	**0.6425**	**1.5439**	**0.5482**	**1.4217**	**0.4726**	**1.3919**	**0.4451**	**1.3506**	**0.4233**
Gains	10.13%	8.76%	9.77%	22.15%	3.30%	3.87%	1.07%	5.85%	0.05%	2.25%

Table 2
Performance comparison on throughput prediction in the high-sparsity situation.

addressesfMethods	Density = 0.5%(D6)		Density = 1%(D7)		Density = 2%(D8)		Density = 3%(D9)		Density = 4%(D10)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
UMEAN [12]	112.1960	55.0068	111.2856	54.4074	110.8732	53.8467	110.7070	53.6448	110.5742	53.1459
IMEAN [13]	90.7025	35.3947	82.0907	32.2714	74.3171	29.0451	72.1900	28.8661	71.3071	28.6747
UPCC [12]	112.0696	54.9263	109.1217	52.6400	106.2907	49.4334	102.9665	47.3301	95.4764	41.5693
IPCC [13]	120.5881	47.5637	120.3505	47.2248	114.1630	46.1030	90.5880	38.6253	73.4479	32.2851
UIPCC [13]	108.0630	51.1348	96.4888	42.1151	81.0827	34.6690	74.6717	33.4364	73.4309	31.6483
NMF [26]	106.7193	43.0964	98.4433	39.4481	90.3144	34.1813	89.1790	33.1333	83.7010	32.5076
HMF [27]	103.6623	39.9637	96.1649	35.8862	92.1575	34.8761	90.0797	34.1184	88.3741	33.4422
LMF-PP [29]	108.3422	39.2184	95.9179	33.1171	77.3738	28.6993	72.1616	26.5359	67.3643	24.9041
LDCF [42]	100.8081	35.1775	83.5723	30.2831	68.5387	23.8206	58.7678	18.6076	55.4478	17.1913
MGCF-DFM [35]	90.7944	33.6987	79.5741	29.2623	71.4496	25.0795	64.9274	23.1045	55.9968	17.6414
BGCL	**87.3661**	**31.8826**	**77.4896**	**28.0268**	**65.0578**	**20.7998**	**56.3769**	**18.0680**	**51.6141**	**15.9830**
Gains	13.33%	9.37%	7.28%	7.45%	5.08%	12.68%	4.07%	2.90%	6.91%	7.03%

D1(0.5%), D2(1%), D3(2%), D4(3%), and D5(4%) were extracted from the dataset on response time. D6(0.5%), D7(1%), D8(2%), D9(3%), D10(4%) were obtained from the dataset on throughput.

Experimental results on response time and throughput are shown in Tables 1 and 2, respectively. To compare the performance between BGCL and LDCF, the best baseline method, the gains are calculated as follows.

$$\text{Gains} = \frac{\text{value}(LDCF) - \text{value}(BGCL)}{\text{value}(LDCF)}, \tag{20}$$

where value(F) indicates the performance of method F.

According to Tables 1 and 2, the following observations can be drawn: (1) Existing approaches produce poor results at low density, despite the fact that observed data is quite sparse in reality scenarios. (2) As data density grows, the performance of all techniques improves, highlighting the necessity of having enough input data for QoS prediction. (3) The model-based collaborative filtering prediction approaches outperform the memory-based ones in terms of prediction accuracy. (4) In most situations, neural network-based approaches (e.g., LMF-PP, LDCF, MGCCF-DFM, and BGCL) outperform memory-based methods and factorization model-based methods in terms of RMSE and MAE, confirming that deep neural networks can learn well-preformed representations for QoS prediction. (5) Graph network-based approaches (e.g., MGCFDFM, BGCL) outperform in low matrix density scenarios, indicating that the graph network may mine implicit information from sparse graph structures. Furthermore, we see that BGCL surpasses MGCF-DFM. (6) In low matrix density scenarios, BGCL achieves the best performance. It has been demonstrated that BGCL can extract high-performing embeddings from sparse data, hence reducing the problem of data sparsity and cold-start.

4.5. Comparison of prediction accuracy for cold-start

To analyze the prediction accuracy of BGCL for the cold-start problem, we randomly selected half of the users in the original dataset as cold-start users, meaning that these users have no invocation records in the training set. The interaction entries were then randomly removed from the remaining QoS matrix with different densities to serve as the training set for BGCL. After removing the selected cold-start users, the dataset was constructed in a similar way to Section 4.4. C1(0.5%), C2(1%), C3(2%), C4(3%), C5(4%) were extracted from the response time dataset. C6(0.5%), C7(1%), C8(2%), C9(3%), C10(4%) were generated from the throughput dataset.

Due to the lack of interaction information for the cold-start users in the training set, UMEAN, UPCC, and UIPCC are unsuitable for this situation; hence the baselines for this experiment no longer include these three methods. Experimental results of response time and throughput are shown in Tables 3 and 4, respectively. The following observations can be made: (1) In the cold-start scenario, almost all methods are less effective than random sampling with all data for training, which highlights the need of improving the cold-start prediction accuracy. (2) BGCL outperforms other baseline approaches in predicting QoS values for cold-start users, hence mitigating the cold-start issue.

4.6. Impact analysis of parameters

The main parameter settings of BGCL include: the ratio of dropout P, the weight of similarity α, the temperature in softmax τ, the number of neighbors N, the embedding dimension of user K. A series of experiments were conducted to investigate the effects of these parameters.

4.6.1. Parameter determination

The hyperparameter P, in BGCL, is the ratio of dropout, which determines the probability of dropping out the edges to generate different perspectives of sub-graphs. Furthermore, α is used to balance the importance of invocation similarity and location

Table 3
Performance comparison on response time prediction in the cold-start situation.

addressesfMethods	Density = 0.5%(C1)		Density = 1%(C2)		Density = 2%(C3)		Density = 3%(C4)		Density = 4%(C5)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
IMEAN [13]	2.0304	0.8394	1.8959	0.7940	1.7978	0.7778	1.7042	0.7560	1.6467	0.7348
IPCC [13]	2.1341	0.8949	2.1628	0.9162	2.1439	0.8811	2.1674	0.8942	2.1063	0.8572
NMF [26]	2.1870	1.6811	2.0554	1.5209	1.9627	1.3770	1.8731	1.2501	1.7652	1.1073
HMF [27]	2.3099	1.5681	2.2771	1.4708	2.2103	1.3305	2.0559	1.2044	1.8546	1.0392
LMF-PP [29]	1.9270	0.8351	1.8980	0.7386	1.8301	0.7238	1.7788	0.6698	1.6853	0.6443
LDCF [42]	1.8864	0.7457	1.8265	0.7129	1.7297	0.6697	1.6688	0.6109	1.6061	0.5655
MGCF-DFM [35]	1.8537	0.8026	1.8012	0.6901	1.7201	0.6005	1.7068	0.5938	1.6132	0.5599
BGCL	**1.8446**	**0.7268**	**1.7798**	**0.6736**	**1.7173**	**0.6414**	**1.6585**	**0.5935**	**1.6004**	**0.5577**
Gains	2.22%	2.54%	2.56%	5.51%	0.72%	4.23%	0.62%	2.84%	0.35%	1.38%

Table 4
Performance comparison on throughput prediction in the cold-start situation.

addressesfMethods	Density = 0.5%(C6)		Density = 1%(C7)		Density = 2%(C8)		Density = 3%(C9)		Density = 4%(C10)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
IMEAN [13]	95.7025	38.3947	87.0907	35.2714	82.3171	29.7254	78.3071	29.0451	72.1900	28.8661
IPCC [13]	120.2908	47.6203	120.0982	47.4832	122.1739	48.5278	120.6128	47.7984	121.9700	48.3595
NMF [26]	108.9999	44.0225	100.5857	40.5097	97.0531	36.8258	90.2544	34.1849	89.3990	33.3104
HMF [27]	107.1734	42.1015	99.9592	37.5555	101.9947	39.6587	99.9919	36.6515	93.3305	34.2158
LMF-PP [29]	111.0314	45.8744	110.2515	39.5038	93.3843	36.6333	86.1568	34.2041	82.6183	30.8995
LDCF [42]	107.0368	39.3950	102.9499	37.0925	89.4251	32.7096	74.9650	27.4545	72.9647	27.1299
MGCF-DFM [35]	98.2656	40.1376	88.6542	34.5040	83.2391	30.9734	75.2470	28.6397	73.6545	28.3025
BGCL	**94.6935**	**37.6694**	**86.0313**	**31.5286**	**80.3634**	**28.6244**	**73.2179**	**26.9627**	**71.4859**	**25.8113**
Gains	11.53%	4.38%	16.43%	15.00%	10.13%	12.49%	2.33%	1.79%	2.03%	4.86%

similarity for obtaining the final similarity ranking. The hyperparameter τ controls the strength of penalties on the hard negative samples. The value of parameter N determines the number of sampled neighbors when GAT aggregates neighbor information. In BGCL, K denotes the dimensionality of user and service embedding. It controls the capacity of the representation model.

We validate a large range of the parameters with a series of experiments. For the different scale levels of datasets, there are small changes in the optimal parameters. We find that a promising performance can be achieved with the parameters $P = 0.2$, $\alpha = 0.7$, $\tau = 0.7$, $N = 6$ and $K = 128$.

4.6.2. Ratio of dropout

The edges were dropped out with a certain ratio P to obtain different perspectives of sub-graphs. To verify the influence of dropout, we varied P from 0.1 to 0.9 and carried out a series of experiments. The performance results in terms of MAE and RMSE on the datasets are plotted in Fig. 3. It can be seen when P increases from 0.1 to 0.2, both MAE and RMSE decrease, indicating an improvement in prediction accuracy. As P continues to increase, both MAE and RMSE maintained a trend of fluctuations. But their MAE and RMSE are both greater than those when P is set to 0.2, indicating a decline in prediction accuracy. We believe that this is due to the fact that employing a certain probability for edge dropout is helpful for building multiple perspectives on sub-graphs, but dropping out too much data causes the existing data to be underutilized, resulting in poor results. Experimental results show that the best results are obtained when P is set to 0.2.

4.6.3. Weight of similarity

The ultimate similarity ranking is calculated by combining invocation and location similarity. Given that geographical location influences invocation similarity, the weight mechanism is used to balance the two similarities. We vary α from 0 to 1 to find a better weight setting. The performance results in terms of MAE and RMSE metrics on the datasets are plotted in Fig. 4. The experimental results indicate that when the weight is set to 0.7, the best prediction effect is achieved. When the weight is set to this value, we believe the two similarities are properly balanced.

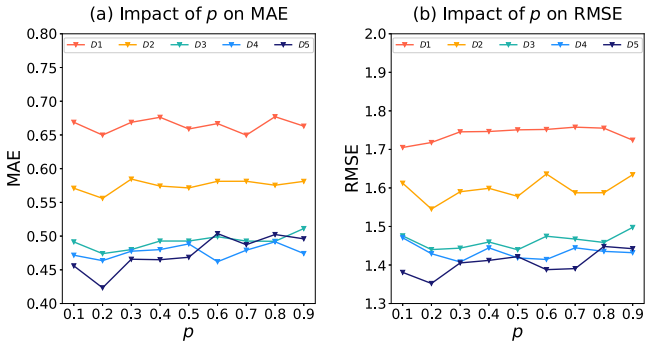

Fig. 3. The impact of the ratio of dropout (P).

4.6.4. Temperature in softmax

The temperature is used to regulate the level of attention to the hard negative samples. Specifically, the contrastive loss with low temperature penalizes considerably more on the harder negative samples, resulting in a more separated local structure of each sample and a more uniform embedding distribution. On the contrary, contrastive loss at high temperature is less sensitive to hard negative samples, and the hardness-aware property disappears gradually as the temperature rises. To find a better temperature setting, we conducted a series of comparative experiments on τ in the range of 0.1 to 1. The performance results are shown in Fig. 5. It can be observed that the best performance is obtained when τ is set to 0.7.

4.6.5. Number of neighbors

In the experiments, we sampled a fixed number of neighbors to obtain node embedding instead of aggregating the information from all the node's neighbors. To analyze the influence of the maximum number of neighbors N, we vary N in the range of {4, 5, 6, 7, 8, 9}. The performance results in terms of MAE and RMSE on the datasets are plotted in Fig. 6. The MAE and RMSE values fall as the number of nodes rises until they reach a particular threshold. It suggests that as the number of neighbors grows,

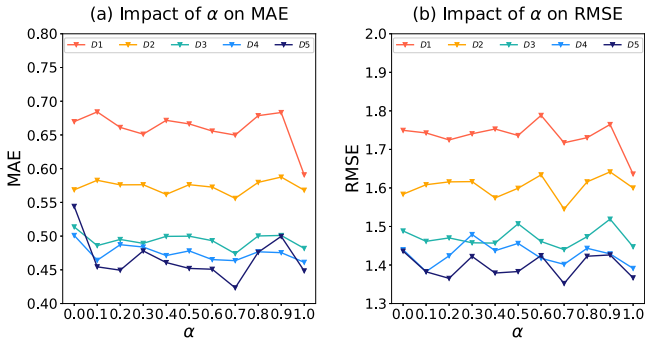

Fig. 4. The impact of the weight of similarity (α).

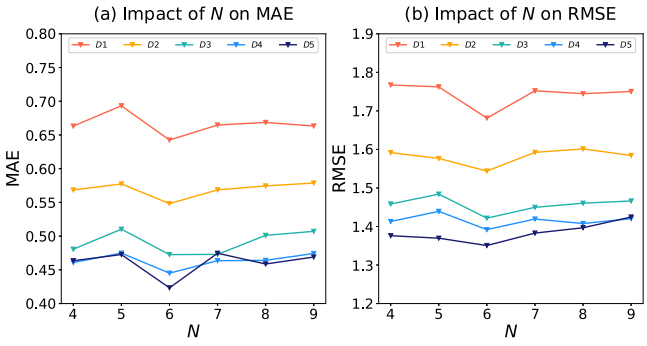

Fig. 6. The impact of the number of neighbors (N).

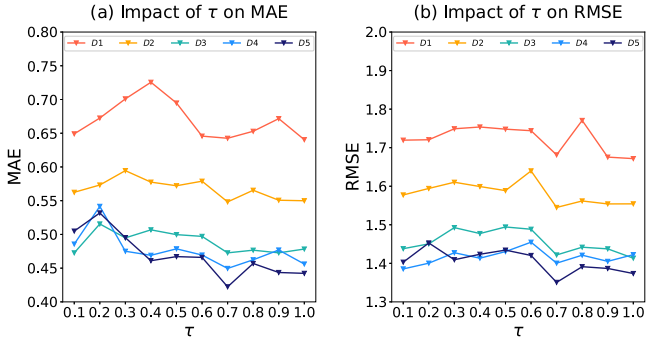

Fig. 5. The impact of the temperature in softmax (τ).

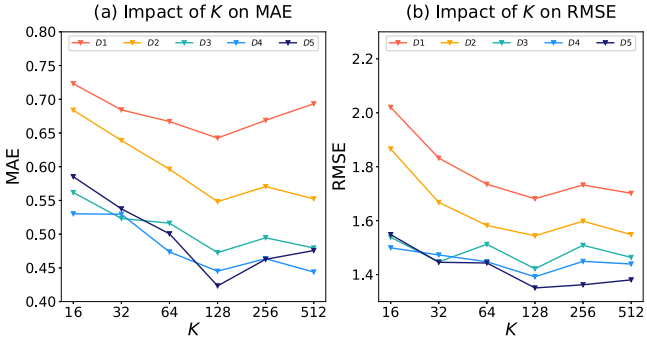

Fig. 7. The impact of the embedding dimension (K).

nodes can gather more information from their neighbors and mine the network structure to learn better representations and obtain more accurate QoS predictions. However, after a threshold is reached, the prediction accuracy of QoS improves very slightly or even decreases as the number of neighbors increases. This is most likely due to weakly correlated neighbors impairing node embeddings, resulting in the learning of poor-performing representations. The results show that when the number of neighbors is 6, the MAE and RMSE produce the best outcomes.

4.6.6. Embedding dimension

The choice of the dimensionality of the high-dimensional space is critical. Lower dimensions will result in overly abstract derived features, while higher dimensions will contain noise, making it impossible to extract precise high-dimensional features. To study the effect of the embedding dimensions of users and services K, we varied K in the range of {16, 32, 64, 128, 256, 512}. The impact analysis result of the parameter on prediction accuracy is shown in Fig. 7. The results indicate that the MAE and RMSE decrease as the embedding size increases until a certain threshold is reached. It indicates that 128 is the optimal dimension size.

4.7. Ablation experiment

To further validate the effectiveness of our model, we conduct some ablation studies.

4.7.1. Advantage of using location information

Geographic location data plays an essential role in poor-performing values. To evaluate the effectiveness of introducing geolocation data, we developed a GCL model. The distinction between GCL and BGCL is that BGCL includes geographic location information. We conducted a comparative experiment on the constructed dataset. The experimental results are shown in Fig. 8. Compared to GCL, the MAE of BGCL on D1, D2, D3, D4, and D5 is increased by 2.81%, 4.54%, 3.61%, 5.70%, 7.70%, while the RMSE is increased by 3.23%, 2.82%, 1.63%, 1.37%, and 2.07%. On average, MAE is improved by 4.87% and RMSE by 2.07%. The experimental results demonstrate that the introduction of geolocation information improves the accuracy of QoS prediction, which confirms our conjecture on the effectiveness of geolocation information.

4.7.2. Advantage of using graph contrastive learning

In QoS prediction, the two most challenging problems to handle are low density and cold-start. Utilizing the existing data for accurate prediction is a crucial area of study given the limited observed interactions. We utilize graph contrastive learning as a data augmentation method for the dataset, building subgraphs from multiple viewpoints based on a limited amount of existing data in order to understand the characteristics of users and services. The final features are obtained by integrating the high-dimensional features learned from the sub-graphs. To demonstrate the effectiveness of graph contrastive learning, we developed the SGCL model. SGCL is equivalent to a single-graph variant of BGCL in which graph comparison learning is omitted. We conducted a comparative experiment on the constructed dataset. The experimental results are shown in Fig. 9. Compared to SGCL, the MAE of BGCL on D1, D2, D3, D4, and D5 is increased by 1.82%, 4.30%, 5.42%, 5.80%, and 8.65%, while the RMSE is increased by 4.98%, 3.35%, 2.34%, 1.12%, and 2.07%. On average, MAE is improved by 5.20% and RMSE by 2.77%. According to the experimental results, the use of graph contrastive learning improves the accuracy of QoS prediction.

5. Conclusion

In this paper, we propose a Web service QoS prediction method based on graph contrastive learning. Firstly, we generate

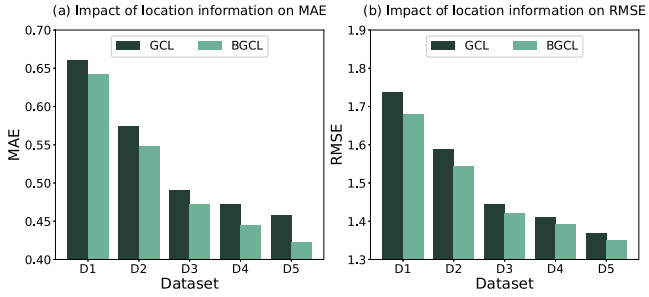

Fig. 8. The impact of location information.

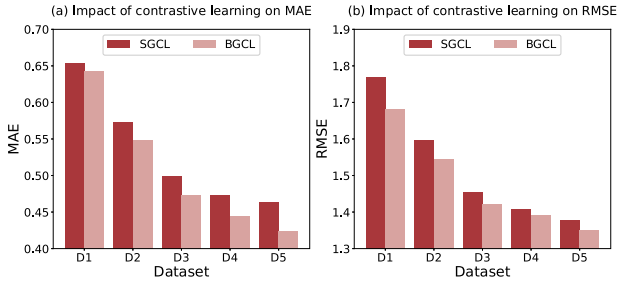

Fig. 9. The impact of graph contrastive learning.

different perspectives of user-neighborhood sub-graphs and service neighborhood sub-graphs based on the sparse user–service bipartite graph. Next, user and service embeddings are learned using graph contrastive learning and graph attention aggregation on the generated sub-graphs. The user and service embeddings are then fed into an MLP to predict QoS values. This strategy makes full use of user–service bipartite network information and user reputation information. Thus it can fully mine the high-order implicit relations between users (or services) from the historical data and discover more similar users for the target users. It not only considers the impact of untrustworthy users on prediction accuracy but also alleviates the problem of data sparsity to some extent. Experiments show that our method is more accurate than the existing methods.

One future direction of this work is to combine the current model and collaborative filtering method to make full use of the advantages of collaborative filtering. Moreover, we will further consider the impact of timing factors on QoS, which includes various factors such as geographic location information and changes in network conditions over time.

CRediT authorship contribution statement

Jiangyuan Zhu: Methodology, Software, Writing – original draft. **Bing Li:** Conceptualization, Writing – review & editing, Supervision. **Jian Wang:** Conceptualization, Methodology, Writing – review & editing. **Duantengchuan Li:** Methodology, Writing – review & editing. **Yongqiang Liu:** Conceptualization, Writing – review & editing. **Zhen Zhang:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 62032016 and 61972292) and the Key Research and Development Program of Hubei Province (No. 2021BAA031).

References

[1] X. Xue, G. Li, D. Zhou, Y. Zhang, L. Zhang, Y. Zhao, Z. Feng, L. Cui, Z. Zhou, X. Sun, et al., Research roadmap of service ecosystems: A crowd intelligence perspective, Int. J. Crowd Sci. 6 (4) (2022) 195–222.
[2] M. Silic, G. Delac, S. Srbljic, Prediction of atomic web services reliability based on K-means clustering, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, 2013, pp. 70–80.
[3] Y. Huo, P. Qiu, J. Zhai, D. Fan, H. Peng, Multi-objective service composition model based on cost-effective optimization, Appl. Intell. 48 (3) (2018) 651–669.
[4] E. Ahmad, M. Alaslani, F.R. Dogar, B. Shihada, Location-aware, context-driven QoS for IoT applications, IEEE Syst. J. 14 (1) (2020) 232–243.
[5] F. Wang, Y. Chiu, C. Wang, K. Huang, A referral-based QoS prediction approach for service- based systems, J. Comput. 13 (2) (2018) 176–186.
[6] S. Ding, C. Xia, Q. Cai, K. Zhou, S. Yang, QoS-aware resource matching and recommendation for cloud computing systems, Appl. Math. Comput. 247 (2014) 941–950.
[7] P. Zhang, H. Jin, H. Dong, W. Song, L. Wang, LA-LMRBF: online and long-term web service QoS forecasting, IEEE Trans. Serv. Comput. 14 (6) (2021) 1809–1823.
[8] B. Cao, X.F. Liu, M.M. Rahman, B. Li, J. Liu, M. Tang, Integrated content and network-based service clustering and web apis recommendation for mashup development, IEEE Trans. Serv. Comput. 13 (1) (2017) 99–113.
[9] D. Yu, Y. Liu, Y. Xu, Y. Yin, Personalized QoS prediction for web services using latent factor models, in: 2014 IEEE International Conference on Services Computing, 2014, pp. 107–114.
[10] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, A collaborative filtering approach to mitigate the new user cold start problem, Knowl.-Based Syst. 26 (2012) 225–238.
[11] Y. Ma, X. Geng, J. Wang, A deep neural network with multiplex interactions for cold-start service recommendation, IEEE Trans. Eng. Manage. 68 (1) (2021) 105–119.
[12] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[13] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, H. Mei, Personalized QoS prediction forweb services via collaborative filtering, in: IEEE International Conference on Web Services (ICWS 2007), 2007, pp. 439–446.
[14] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 426–434.
[15] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.
[16] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, G. Chen, Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems, in: The World Wide Web Conference, 2019, pp. 2091–2102.
[17] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, X. Xie, Self-supervised graph learning for recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 726–735.
[18] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[19] J. Bennett, S. Lanning, et al., The netflix prize, in: Proceedings of KDD Cup and Workshop, Vol. 2007, 2007, p. 35.
[20] Q. Li, S.H. Myaeng, B.M. Kim, A probabilistic music recommender considering user opinions and audio features, Inf. Process. Manage. 43 (2) (2007) 473–487.
[21] A. Nocera, D. Ursino, An approach to providing a user of a "social folksonomy" with recommendations of similar users and potentially interesting resources, Knowl.-Based Syst. 24 (8) (2011) 1277–1296.
[22] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, M.R. Lyu, Web service QoS prediction via collaborative filtering: A survey, IEEE Trans. Serv. Comput. (2020).
[23] X. Luo, J. Liu, D. Zhang, X. Chang, A large-scale web QoS prediction scheme for the industrial internet of things based on a kernel machine learning algorithm, Comput. Netw. 101 (2016) 81–89.
[24] Z. Zheng, H. Ma, M.R. Lyu, I. King, QoS-aware web service recommendation by collaborative filtering, IEEE Trans. Serv. Comput. 4 (2) (2011) 140–152.

[25] Z. Zheng, M.R. Lyu, Collaborative reliability prediction of service-oriented systems, in: 2010 ACM/IEEE 32nd International Conference on Software Engineering, Vol. 1, IEEE, 2010, pp. 35–44.
[26] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[27] P. He, J. Zhu, Z. Zheng, J. Xu, M.R. Lyu, Location-based hierarchical matrix factorization for web service recommendation, in: 2014 IEEE International Conference on Web Services, 2014, pp. 297–304.
[28] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, T. Zhang, Collaborative web service quality prediction via exploiting matrix factorization and network map, IEEE Trans. Netw. Serv. Manag. 13 (1) (2016) 126–137.
[29] D. Ryu, K. Lee, J. Baik, Location-based web service QoS prediction via preference propagation to address cold start problem, IEEE Trans. Serv. Comput. 14 (3) (2021) 736–746.
[30] I. Maksimov, R. Rivera-Castro, E. Burnaev, Addressing cold start in recommender systems with hierarchical graph neural networks, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 5128–5137.
[31] R. van den Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, 2017, CoRR abs/1706.02263.
[32] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, 2019, pp. 165–174.
[33] Y. Li, J. Xu, W. Liang, GraphMF: QoS prediction for large scale blockchain service selection, in: 2020 3rd International Conference on Smart BlockChain (SmartBlock), 2020, pp. 167–172.
[34] L. Ding, G. Kang, J. Liu, Y. Xiao, B. Cao, QoS prediction for web services via combining multi-component graph convolutional collaborative filtering and deep factorization machine, in: 2021 IEEE International Conference on Web Services, ICWS, 2021, pp. 551–559.
[35] Z. Chang, D. Ding, Y. Xia, A graph-based QoS prediction approach for web service recommendation, Appl. Intell. 51 (10) (2021) 6728–6742.
[36] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020 pp. 9729–9738.
[37] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.
[38] K. Zhou, H. Wang, W.X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, J.-R. Wen, S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1893–1902.
[39] T. Yao, X. Yi, D.Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E.H. Chi, S. Tjoa, J. Kang, et al., Self-supervised learning for large-scale item recommendations, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 4321–4330.
[40] T. Chen, S. Kornblith, M. Norouzi, G.E. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, in: Proceedings of Machine Learning Research, vol. 119, 2020, pp. 1597–1607.
[41] Z. Zheng, Y. Zhang, M.R. Lyu, Distributed QoS evaluation for real-world web services, in: 2010 IEEE International Conference on Web Services, 2010 pp. 83–90.
[42] Y. Zhang, C. Yin, Q. Wu, Q. He, H. Zhu, Location-aware deep collaborative filtering for service recommendation, IEEE Trans. Syst. Man Cybern. 51 (6) (2021) 3796–3807.