Neurocomputing 549 (2023) 126429

Contents lists available at ScienceDirect

Neurocomputing

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

Neighbor importance-aware graph collaborative filtering for item recommendation

Qingxian Wang [a], Suqiang Wu [a], Yanan Bai [b], Quanliang Liu [c], Xiaoyu Shi [c,*]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu 610054, China
[b] School of Artificial Intelligent, Chongqing University of Technology, Chongqing 401135, China
[c] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China

A R T I C L E I N F O

Article history:
Received 13 August 2022
Revised 17 February 2023
Accepted 4 June 2023
Available online 15 June 2023
Communicated by Zidong Wang

Keywords:
Graph neural networks
Recommender system
Node importance
Collaborative filtering
Representation learning

A B S T R A C T

The emerging topic of Graph Neural Networks (GNN) has attracted increasing attention and achieved state-of-the-art (SOTA) performance in many recommendation problems, due to its strong ability in node representation with exploring high-order information. To learn a node's representation, previous methods usually linearly combine the embeddings of node features, amusing the equal importance of neighbors. However, due to the intrinsic differences (i.e., degree, create time) over neighbors, we argue that these differences carry important signals for node representation. Ignoring them will lead to a suboptimal in node representation and thus weaken the effectiveness of the follow-up graph-based operations. To address it, we propose BIG-SAGE@ for item recommendation with rating prediction task, which is a neighbor importance-aware graph neural network. Specifically, its main idea is twofold: 1) A rating confidence-based neighborhood sampling method is introduced, making the sampling process biased to those more valuable nodes. 2) An attention network is integrated to achieve the rating prediction task, by flexibly incorporating information from user and item embedding features. Finally, we verified the effectiveness of the proposed model on six public data sets. Extensive experimental results demonstrate the superior performance of BIG-SAGE@ in the rating prediction and TopN ranking tasks, compared to the SOTA methods.

1. Introduction

As one of the most powerful tools to alleviate the information overloading issue, recommender system (RS) has been widely developed to perform personalized information recommendation [1,2]. Compared to the search engine, the core of RS is to predict the rating or preference a user would give to an item. Intrinsically, collaborative filtering (CF) [3–6], which focuses on exploiting the historical user-item interactions to predict the next-step interaction, remains an essential yet fundamental method to achieve effective personalized recommendation.

The most common stages of CF are: 1) embedding, which learns low-rank latent factors to represent a user and an item, and 2) interaction modeling, which is constructed based on the embedding vector of the user and item. As a pioneer, matrix factorization (MF) directly embeds the user/item ID into a low-rank latent feature, then makes a prediction based on the inner product of latent features [7,8]. To better exploit the rich user/item side information, collaborative deep learning integrates the rich side information of the user and item into the MF component with deep representation [9]. Moreover, neural collaborative filtering (NCF) model was proposed to replace the MF interaction function of the inner product with a nonlinear neural network [10]. To improve the modeling ability of user-item interaction, a deep factorization machine (DeepFM) combined the shallow model FM with multiple-layer perceptions (MLP)[11]. xDeepFM is an improved version that combines the explicit and implicit feature interactions to improve its capacity [12]. However, these methods still have improvement space, since their paradigms of prediction and training ignore the high-order structural information in observed data.

Recently, with the success of graph neural networks (GNN) on graph structure data, several CF methods utilize graph neural networks to improve the recommendation performance, since the user-item interactions can be modeled as a natural user-item interaction graph (i.e., bipartite graph) [13–17]. Based on the bipartite graph, GNN can aggregate neighborhood embedding iteratively via embedding propagation. Thus, each node can access the high-order neighbors' information, rather than only the first-order neighbors', as the traditional methods do. With its advantages in

* Corresponding author.

https://doi.org/10.1016/j.neucom.2023.126429
0925-2312/© 2023 Elsevier B.V. All rights reserved.

handling structural data and exploring structural information, GNN-based collaborative filtering methods have become the new state-of-the-art (SOTA) approach in recommender systems.

Although the GNN-based method has shown superior performance, its performance efficiency is rather heavy - aggregating and updating multi-hop neighbors. As a result, the existing GNN-based methods usually suffer the neighbor explosion problem, when dealing with the large-scale graph structure (e.g., RS). In RS, the user-item interaction graphs are large-scale, often containing millions to billions of entities (i.e., users and items). In such a case, it requires extensive computation costs for aggregating and updating multi-hop neighbors, due to their size and complexity.

To address this issue, several node sampling methods have been proposed to improve the training efficiency of GNN-based methods. Most GCF methods adopt random or random walk-based sampling on the previous layer neighbors to obtain a small number of neighbors to perform the aggregation process. As a result, the recommendation performance has a certain loss, due to the important difference among the neighborhood being seriously ignored. Thus, we argue that there is much room for improvement.

Though appealing, the difficulty of our work is that GNN was originally proposed for node classification on the attributed graph, where each node has rich attributes as input features. However, the CF task has no rich attributes as input features. Regarding the user-item interaction graph of CF, each node (user or item) is only described by a one-hot ID, which has no concrete semantics besides being an identifier. In such a case, given the ID embedding as the input, how to build an efficient neighbor sampling method without sacrificing recommendation accuracy is one of the vital keys to the success of our study.

To this end, we present BIG-SAGE@, a bipartite graph-based sample and aggregate method for large-scale dynamic interactive networks, where the symbol @ means an attention mechanism is integrated for item recommendation. To address the above issue, in the embedding state, we select two important yet easily obtained signals (i.e., timestamp and node degree) with clear physical meaning to evaluate the node importance. To be specified, the timestamp reflects the timeliness of the interaction between nodes, which can be commonly observed in the CF task[18–20], while the node's degree reflects the node's role in maintaining the connectivity of the graph. In the interaction modeling stage, the existing GCF methods usually adopt the dot product between user and item embedding to make rating predictions. However, we argue that the naive dot product operation of user and item embedding can not distinguish the important difference between each dimension. Based on the above analysis, the essential ideas are threefold: 1) Building a rating confidence function to evaluate the importance of each neighbor node; 2) Performing the state aggregating and updating based on the sampled neighbors during the embedding stage; 3) Leveraging the attention mechanism to identify the contributions of each dimension of user and item embeddings during the interaction modeling stage. To summarize, the main contributions of this study include:

- We propose a novel and efficient graph collaborative filtering (i.e.,BIG-SAGE@) for the large-scale item recommendation, where this target can be achieved by identifying valuable neighbors and embedding dimensions.
- We design a rating confidence-based neighbor sampling method with an explicit function to improve training efficiency, by exploring timestamp and node degree as input. The most significant benefit is that it has an explanation ability, compared to the complex neural network-based methods. Furthermore, we integrate the attention model to flexibly model the importance of embedding features in the interaction modeling stage.
- We conduct extensive experiments on rating prediction and TopN tasks on several industrial datasets. The experimental results demostrate that the BIG-SAGE@ achieves competitive and superior results on rating prediction and TopN rank tasks, compared to SOTA methods.

The rest of this paper is arranged as follows: Section 2 gives the preliminaries. Section 3 states the BIG-SAGE@ model. Section 4 gives the experimental results and analyses. Finally, Section 5 discusses and concludes this paper.

2. Preliminaries

2.1. Problem formulation

We first introduce the necessary definition and notations in this paper, as shown in Table 1. Specifically, we adopt bold capital letters (e.g.,$\mathbf{X}$) and bold lowercase letters (e.g., $\mathbf{x}$) to denote matrices and vectors, respectively. x_{ij} represents the entry of matrix $\mathbf{X}$ in the row i and column j. Moreover, all vectors are in column form if not otherwise specified.

We then describe the graph-based collaborative filtering problem as follows: In RS, we usually have M users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$, and a set of N items $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. In CF-based recommendation scenario, the content of a user-item interaction generally includes a four-element tuple: $< u, v, r_{uv}, t_{uv} >$, where u and v are the user ID and item ID, r_{uv} represents user u has rated on item v, and t_{uv} records the when the user-item interaction happen. We also denote a user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{M\times N}$, where $y_{uv} = 1$ is that user u has rated on item v(i.e., $r_{uv} > 0$), otherwise $y_{uv} = 0$. Then the user-item interactions can be naturally represented as a user-item bipartite graph (i.e., rating graph). $\mathcal{G} =< U \cup \mathcal{V}, \mathcal{A} >$, where $\mathcal{A} \in \mathbb{R}^{(M+N)\times(M+N)}$ is the adjacency matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{Y} \\ \mathbf{Y}^T & 0 \end{pmatrix} \tag{1}$$

Thus, our work aims to learn the user and item representations and then estimate the degree of user preference for an item based on the learned user and item representations.

2.2. Graph neural networks

With the rapid emergence of vast volumes of graph data, e.g., social networks and molecular structures, a wave of graph neural network studies has sprung up in recent years [21–25]. GNN aims to aggregate neighbor state information and update each node layer by layer. The general framework of graph neural networks mainly focuses on state aggregation and updates in the model. In this study, we use GraphSAGE [26] as a reference to describe the graph neural network framework symbolically. In GraphSAGE, the process of aggregating the state representation of neighbor nodes can be expressed:

$$\mathbf{h}^{(l)}_{\mathcal{N}(i)} = AGG^{(l)}\left(\left\{\mathbf{h}^{(l-1)}_j, \forall j \in \mathcal{N}(i)\right\}\right) \tag{2}$$

where $\mathbf{h}^{(l-1)}_j$ is the state representation of the neighbor node j at the $(l-1)$-th level. $AGG^{(l)}(\cdot)$ is the aggregation operation in the l-th layer, which is a core component in GNN. The implementation schemes of $AGG^{(l)}(\cdot)$ can refer to the following Table 2.

Based on the aggregation results of neighbors, the updating process of target node i is:

$$\mathbf{h}^{(l)}_i = \sigma\left(\mathbf{W}^{(l)}\left[\mathbf{h}^{(l-1)}_i \| \mathbf{h}^{(l)}_{\mathcal{N}(i)}\right]\right) \tag{3}$$

Table 1
Notations and definitions used in this paper

Symbols	Description
$\mathcal{G}$	A graph (i.e., the rating graph)
$\mathbf{V}$	Node set of $\mathcal{G}$
$\mathbf{A} \in \mathbb{R}^{\lvert\mathbf{X}\rvert \times \lvert\mathbf{X}\rvert}$	The adjacency matrix of a graph
$\mathcal{U} \in \mathbb{R}^{M}, \mathcal{V} \in \mathbb{R}^{N}$	Entity set of users and items with size of M and N, respectively
$\mathbf{R}, \widehat{\mathbf{R}} \in \mathbb{R}^{M \times N}$	Target rating matrix and its approximation
Λ, Γ	Known and unknown entry set of $\mathbf{R}$
$\mathbf{S}, \mathbf{T} \in \mathbb{R}^{M \times N}$	The timestamps matrix and its normalized form corresponding to $\mathbf{R}$
$r_{uv}, \hat{r}_{uv}$	Single elements in $\mathbf{R}$ and $\widehat{\mathbf{R}}$
s_{uv}, t_{uv}	Single elements in $\mathbf{S}$ and $\mathbf{T}$
$\mathcal{N}(i)$	Neighbors of a node i
$\widetilde{\mathcal{N}}(i)$	Neighbors of a node i via importance-confidence sampling
$\mathbf{h}_i^l$	The embedding of node i in the l-th propagation layer
n	The size of neighbor set
$\odot$	Element-wise product
$\vert\vert$	Concatenate operation
L	The layers of graph neuron network
k	Output dimension of hidden state representations
σ	A non-linear activation function

Table 2
The implementation scheme of aggregate in GNN.

Name	Details
Weighted sum	$\mathbf{h}_{\mathcal{N}(i)}^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}((i))} \mathbf{W}^{(l)} \mathbf{h}_j^{l-1}\right)$
Gated neural unit aggregation	$\mathbf{h}_{\mathcal{N}(i)}^{(l)} = GRU\left(\left\{\mathbf{h}_j^{(l-1)}, \forall j \in \mathcal{N}(i)\right\})\right)$ or
	$\mathbf{h}_{\mathcal{N}(i)}^{(l)} = LSTM\left(\left\{\mathbf{h}_j^{(l-1)}, \forall j \in \mathcal{N}(i)\right\})\right)$
Attention aggregation	$\alpha_j^l = \frac{exp\left(LeakyReLU\left(\alpha^T \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)}\right)\right)}{\sum_{k \in \mathcal{N}(i)} exp\left(LeakyReLU\left(\alpha^T \mathbf{W}^{(l)} \mathbf{h}_k^{(l-1)}\right)\right)}$, $\mathbf{h}_{\mathcal{N}(i)}^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_j^l \mathbf{W}^l \mathbf{h}_j^{k-1}\right)$

where $\mathbf{h}_i^{l-1}$ is the state representation of target node i at $(l-1)$-th layer, $\mathbf{W}^{(l)}$ is a learnable parameter matrix. $\sigma(\cdot)$ is the sigmoid activation function.

In RS, the generation process of user and item representation via GraphSAGE can be described as follow. Take l-th layer for example, for each user node u, it first aggregates its neighbors' representation $\left\{\mathbf{h}_v^{(l-1)}, \forall v \in \mathcal{N}(u)\right\}$ to a single vector $\mathbf{h}_{\mathcal{N}(u)}^{(l)}$ via (4). After that, it then concatenates the user node u's current representation, $\mathbf{h}_u^{(l-1)}$, with the aggregated neighborhood vector, $\mathbf{h}_{\mathcal{N}(u)}^{(l)}$, and feds this concatenated vector into full connect layer. As a result, the user node u's representation at l-th layer can be obtained via (5). Similar to the item node, the l-th representation of item v can be obtained by (6) and (7).

$$\mathbf{h}_{\mathcal{N}(u)}^{(l)} = AGG\left(\left\{\mathbf{h}_v^{(l-1)}, \forall v \in \mathcal{N}(u)\right\}\right) \tag{4}$$

$$\mathbf{h}_u^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \left[\mathbf{h}_u^{(l-1)} \vert\vert \mathbf{h}_{\mathcal{N}(u)}^{(l)}\right]\right) \tag{5}$$

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = AGG\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in \mathcal{N} v\right\}\right) \tag{6}$$

$$\mathbf{h}_v = \sigma\left(\mathbf{W}^{(l)} \cdot \left[\mathbf{h}_v^{(l-1)} \vert\vert \mathbf{h}_{\mathcal{N}(v)}^{(l)}\right]\right) \tag{7}$$

where $\mathcal{N}(u)$ and $\mathcal{N}(v)$ denote the neighbor set of user u and item v, σ is the nonlinear activation function.

2.3. Related work

Graph neural networks: As a solution to graph representation learning, Graph Neural Networks (GNN) have aroused comprehensive scholars' concern in recent years [27–29]. From a broad perspective, GNN refers to all neural networks that accept the input of graphs, including unsupervised deep auto-encoders. The current development of GNN mainly benefits from Graph Convolutional Neural Network (GCN) [30] on a spatial domain and related research. Essentially, Gated Graph Neural Networks (GGNN) [31] and Graph Attention Networks (GAT) [32] both belong to the category of GCN. The difference between these models lies in the method of achieving node status updates. Specifically, the state update function in a graph neural network is to aggregate adjacent nodes' information. In GraphSAGE [26], the state update function is described as an aggregator, which most commonly depends on a weighted average. GGNN is based on gated cyclic units to achieve aggregation, while GAT further introduces the attention mechanism in GCN. GAT adopts the self-attention strategy to better capture neighbor features by dynamically adjusting edge weights. However, in GAT, all neighbors are attended to learn the target node representation. Thus, its training cost is high when dealing with large-scale graph data. GraphSAGE adds hierarchical neighborhood sampling to the original GCN framework compared to GAT. Therefore, it can be better applied to large-scale graph structures. In summary, these graph neural network models use state update methods with different characteristics and add sampling strategies to save memory for large-scale graph learning tasks, which provide some choices for the differentiated requirements of the recommendation scenarios, and a theoretical basis for the application in the recommender system.

GNN-based RS: In the recommender system, the interaction between the user and item can be naturally modeled as a heterogeneous rating graph with two types of nodes [33,34]. Due to the heterogeneity between users and items in the rating graph, some recommendation methods based on GNN generate node representations for users and items, respectively, then use the dot product results of the two as the predicted rating. Recent works show that when GNN is used to obtain better user/item representation, the performance of the recommender can be improved as the collaborative signal and heterogeneous side information is introduced in the form of a graph [35–38]. PinSAGE [37] uses the random-walk method to sample nodes and selects the top *N*neighbors as the neighborhood of the target node. Furthermore, it dynamically constructs a new subgraph structure to perform local graph convolution, and uses minibatch for distributed gradient training. From the perspectives of users and items, NGCF [38] introduces messages that use the weighted average of the neighborhood's initial state as the initial message, and the unweighted average propagates the message in the neighborhood structure layer by layer. Then, the messages in each layer are concatenated as the final node representation. To solve the data sparsity issue in RS, Multi-GCCF [36] utilizes the similarity of the node neighborhood in the user-item rating graph to construct the user-user graph and the item-item graph, and then aggregates the neighborhood information on these two homogeneous graphs to generate the node representation of users and items.

Recently, attention mechanisms have been widely applied in many domains, such as object detection[39], prediction [40], and so on. Several attention-based methods have been proposed for the session-based recommendation to handle the noise or irrelevant interactions in user behavior sequences. These attention-based methods can emphasize those relevant yet essential interactions in a session, while filtering out irrelevant information to the next interaction prediction[41,42]. Thus, we also employ the attention model to enhance recommendation performance. Unlike current attention-based recommendation methods, we focus on learning the user and item representation via a node importance-aware graph neural network, then adopt the attention model to achieve the recommendation task.

Neighborhood sampling methods: Graph structures are mostly large and complex in the real world, and sampling methods can effectively relieve the computational pressure of graph neural networks. Besides, repeated sampling can also prevent nodes from being insufficiently trained for these nodes with sparse connections due to missing data. Therefore, neighbor sampling is an important method for particular graph learning-based applications. It aims to sample a neighborhood structure for each target node, instead of treating all neighbors as its neighborhood structure [43]. To deal with the large-scale graph learning task, several studies focus on the graph sampling method [44,45]. ClusterGCN [44] and GraphSAINT [45] are two representative methods of this kind of research in graph sampling. The basic idea of graph sampling is that it samples the complete training graph to a subgraph and then builds a full GCN on the sampled subgraph. Specifically, GraphSAINT proposes the normalization method to reduce the bias introduced by minibatch estimation, and designs three sampling methods for variance reduction. However, it heavily depends on the graph topology structure to evaluate the node importance, which is not suitable for the recommender system. In the recommender system, the timestamp is an important signal when designing the recommendation model, which can reflect the dynamic change in user preferences. Another line of research focuses on layer sampling. The principle of layer sampling methods follow the three main step: 1) Build a complete GNN on the full training graph; 2) Sample nodes or edges of each layer; 3) Training the node representation on the sampled GNN. Steps 2) and 3) are processed iteratively until algorithm convergence. GraphSAGE [26] adopts the random sampling on the previous layer to create a neighbor set with a small size, then perform an aggregation process among the sampled graph. S-GCN [46] further restricts neighborhood size to only two nodes during sampling on the previous layer. FastGCN[47] performs node sampling independently for each layer. However, they all adopt the random sampling method to construct the neighborhood, and the accuracy of node representation is hardly ensured. Suppose that the number of samples is n. The principle of random sampling is that: if a node's neighbors are less than n, the sampling method with replacement is adopted until n nodes are sampled. Otherwise, sampling without replacement is used for repeated sampling. For the target node u, the set of its neighbors structure is $\mathcal{N}(u)$, then the probability of sampling the neighbor v is:

$$p(v|u) = \frac{1}{\|\mathcal{N}(u)\|}, v \in \mathcal{N}(u) \tag{8}$$

3. Methodology

3.1. The framework of BIG-SAGE@

The objective of BIG-SAGE@ is to improve node representation via explicitly exploiting the important neighbors in a lightweight and efficient manner, and achieves the rating prediction task by flexibly modeling the importance of embedding latent features, whereas the ideal can also be generalized to GNNs for other tasks such as ranking task, community detection[48], node classification [49] and link prediction [50]. BIG-SAGE@ mainly includes three components: rating-confidence based neighbor sampling, state aggregation and update, and attention-based rating prediction. A schematic diagram of the BIG-SAGE@ is presented in Fig. 1, which shows the rating prediction process between target user node u and item node v. Specifically, for each user u, we first adopt the one-hot vector as its initial state $\mathbf{h}_u^{(0)}$ (i.e., static features), and select neighbor nodes $\widetilde{\mathcal{N}}(u)$ of the target user node through the proposed rating-confidence based sampling method. Based on the sampled neighbors, we learn a user representation $\mathbf{h}_u^{(l)}$ by utilizing the graph neural network with state aggregating and update operations. Likewise, each item v is encoded as a representation vector $\mathbf{h}_v^{(l)}$ via the same operations over initial state $\mathbf{h}_v^{(0)}$. After that, $\mathbf{h}_u^{(l)}$ and $\mathbf{h}_v^{(l)}$ are contacted together as input and output the predicted rating $\tilde{r}_{u,v}$ via attention-based interaction model.

The status update processing of the user and item in the GNN have been performed alternately. Moreover, each layer's output should be spliced with the node's initial state, then mapped to the specified output dimension via the parameter matrix, and finally output to the next layer. A more detailed description of BIG-SAGE@ is given as follows.

3.2. Rating-confidence based neighborhood sampling

Inspired by the effectiveness of GNN models in recommendation tasks, BIG-SAGE@ emphasizes enhancing the quality of node representation $\mathbf{h}$ via the proposed rating-confidence based neighborhood sampling method. To avoid important nodes being ignored in the sampling phase, we aim to implement a biased neighbor sampling method instead of uniform neighbor sampling. Our method mainly measures the importance of nodes based on the information provided by the existing connections between nodes. To this end, we begin by defining rating confidence as follows:

Definition 1 (*Rating confidence*). Given the neighbor node v_i and the center node v_j in a rating graph. Assume that there is a edge between v_i and v_j, and the degree of v_j is d_j. For the center node v_j, the rating confidence score between v_j and v_i is c_{ij}, which can be computed as:

$$c_{ij} = e^{\frac{d_i}{1-\lambda t_{ij}}} \tag{9}$$

where:

$$t_{ij} = \frac{s_{ij} - \min(\mathbf{S})}{\max(\mathbf{S}) - \min(\mathbf{S})} \tag{10}$$

In (9), e is a natural constant, t_{ij} indicates the normalized timestamp of the rating r_{ij} between the source node i and target node j. In (10), $\mathbf{S}$ is the timestamps set, s_{ij} represents the timestamp of the rating r_{ij}. λ is used for trading off the importance of degree information and time information ranging in [0,1]. For the normalized operation, the *Min–Max* normalization method is used here.

According to (9), the physical meaning of rating confidence score can be explained that: regardless of whether the source node is a user or an item, the normalized timestamp can indicate the timeliness of the rating behavior, the larger the normalized timestamp, the closer the rating happened to the current time. On the other hand, the degree of the target node indicates whether the node plays an important role in maintaining the graph's connectivity. According to the above definition, the rating confidence between connected nodes can be measured by the rating time and the degree of the node in the rating graph.

Fig. 2 shows the process of computing the rating confidence with the item node v as the center node. The dotted line in the figure is the neighbor structure of the center node, which is also the sampling range. Intuitively, we can build a graph with the rating confidence as the edge via (9), and our sampling is based on this graph.

For the center node v, its neighbors set is $N(v)$, then the probability of sampling a neighbor node u is:

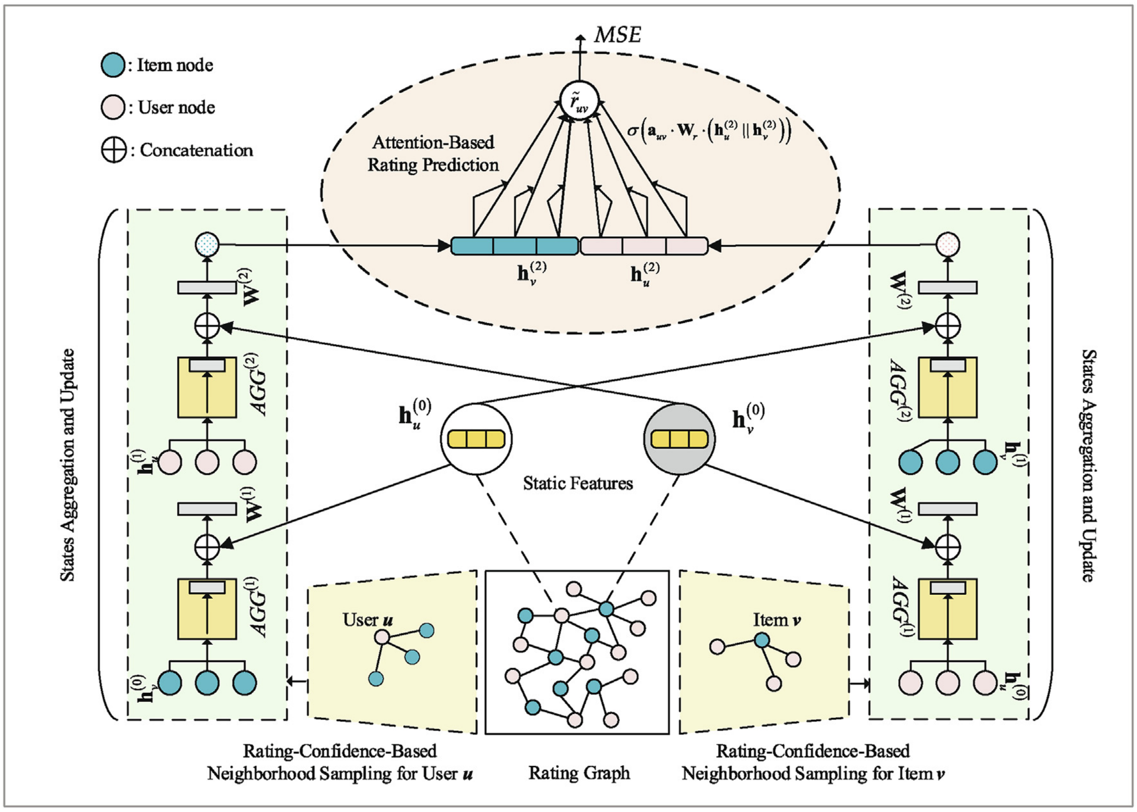


Fig. 1. The BIG-SAGE@ framework.

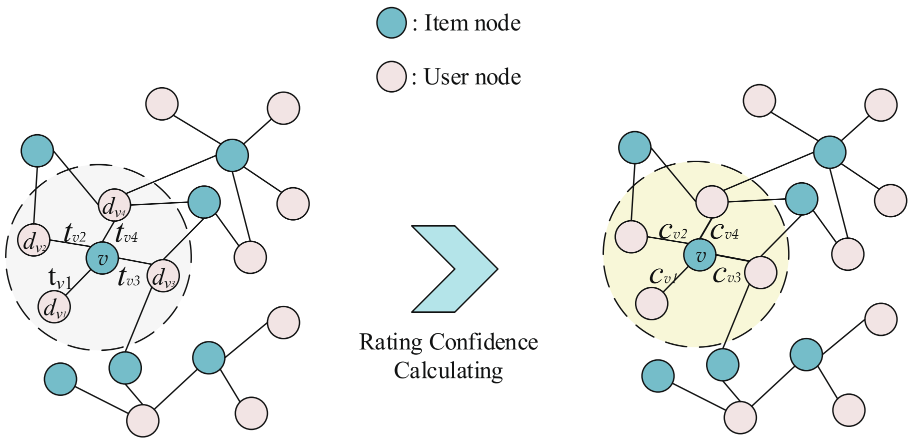


Fig. 2. Computing rating confidence scores from rating graph.

$$p(u|v) = \frac{c_{vu}}{\sum_{i\in\mathcal{N}(v)} c_{vi}}, u \in \mathcal{N}(v) \tag{11}$$

Suppose the size of rating-confidence samples is $n = \widetilde{\mathcal{N}}(v)$. If the number of neighbors is less than n, we use the sampling method with replacement until n nodes are sampled. Otherwise, we apply sampling without replacement. Note that the sample number by the rating-confidence method is less than the size of the neighbors. i.e., $n \ll |N(v)|$.

3.3. States aggregation and update

In RS, the user and item are generally embedded with one-hot vectors by using user and item IDs, whose dimension is equal to the size of the item set. It can be defined as $\mathbf{e}_i = \mathbf{E}\cdot\mathbf{x}_i$, where $\mathbf{E}\in\mathbb{R}^{(M+N)\times d_0}$ is the initial embedding table for users and items. $\mathbf{x}_i$ is the one-hot vector of a userID or an itemID, $\mathbf{e}_i$ is the corresponding embedding. d_0 is the dimension of initial embedding.

In this work, we adopt the weighted summation scheme to implement the aggregation and update of the state, which is shown in Table 2. We describe the state aggregation and update from the user side as an example. Suppose that the target user node is u. $\widetilde{\mathbf{N}}(u)(\|\widetilde{\mathbf{N}}(u)\| = n)$ is the neighbors of user node u after rating-confidence sampling. $\mathbf{h}_u^{(l)}$ represents the state representation of the target user node u in the L-layer, and $\mathbf{h}_u^{(0)} = \mathbf{e}_u$ means the user's initial features. Then the implementations of node aggregation and update are:

$$\mathbf{h}_{\widetilde{\mathbf{N}}(u)}^{(l)} = AGG\left(\left\{\mathbf{h}_{v}^{(l-1)} | v \in \widetilde{\mathbf{N}}(\mathbf{u})\right\}\right) = \sigma\left(\sum_{v \in \widetilde{\mathbf{N}}(u)} \widetilde{\mathbf{W}}_{u}^{(l)} \mathbf{h}_{v}^{(l-1)}\right) \tag{12}$$

$$\mathbf{h}_{u}^{(l)} = \sigma\left(\mathbf{W}_{u}^{(l)} \cdot \left[\mathbf{h}_{u}^{(0)} \| \mathbf{h}_{\widetilde{\mathbf{N}}(u)}^{(l)}\right]\right) \tag{13}$$

where (12) is the aggregator from the user's perspective, it uses a weighted summation scheme to aggregate the state representations of neighbor item nodes in the $(l-1)$-th layer. (13) is a state updater. The output of the aggregator $\mathbf{h}_{\widetilde{N}(u)}^{(l)}$ is concatenated with the state representation of the center node itself $\mathbf{h}_{u}^{(0)}$(i.e., the static feature of the user), and input to the fully connected layer. Then the state representation of a user node in the current layer $h_{u}^{(l)}$ is updated automatically. The final user representation is the L-th layer output $h_{u}^{(L)}$, i.e. ($e_{u}^{*} = \mathbf{h}_{u}^{(L)}$).

Similarly, suppose the center item node is $v \in \mathcal{V}$, and the neighbors of v is $\widetilde{\mathbf{N}}(v)(\|\widetilde{\mathbf{N}}(v)\| = n)$. $\mathbf{h}_{v}^{(l)}$ represents the state representation of the target item node v in the l th-layer, and $\mathbf{h}_{v}^{(0)}$ means the item v's static features (i.e., $\mathbf{h}_{v}^{(0)} = \mathbf{e}_{v}$). The implementation of aggregating and updating item nodes can be defined as follows:

$$\mathbf{h}_{\widetilde{\mathbf{N}}(v)}^{(l)} = AGG\left(\left\{\mathbf{h}_{u}^{(l-1)} | u \in \widetilde{\mathbf{N}}(v)\right\}\right) = \sigma\left(\sum_{u \in \widetilde{\mathbf{N}}(v)} \widetilde{\mathbf{W}}_{v}^{(l)} \mathbf{h}_{u}^{(l-1)}\right) \tag{14}$$

$$\mathbf{h}_{v}^{(l)} = \sigma\left(\mathbf{W}_{v}^{(l)} \cdot \left[\mathbf{h}_{v}^{(0)} \| \mathbf{h}_{\widetilde{\mathbf{N}}_{\mathbf{v}}}^{(l)}\right]\right) \tag{15}$$

where (14) is the aggregator from the item's perspective. It also uses a weighted summation method to aggregate the state representations of the upper layer of neighbor user nodes. (15) is a state updater. The output of the aggregator $\mathbf{h}_{\widetilde{N}(v)}^{(l)}$ is concatenated with the state representation of the target item node itself h_{v}^{0}(static feature of the item), and input to the fully connected layer. Then the item node state representation of the current layer $\mathbf{h}_{v}$ will update. Finally, the node state output of the L-th layer $\mathbf{h}_{v}^{(L)}$ is the final item representation (i.e., $\mathbf{e}_{v}^{*} = \mathbf{h}_{v}^{(L)}$). $\widetilde{\mathbf{W}}^{(l)}$ and $\mathbf{W}^{(l)}$ are the learnable parameter matrix.

3.4. Attention Based Rating Prediction Model

To distinguish the different levels of latent features for the rating prediction task, we employ attenation mechanism to filter out the irrelevant features, which assign different weights to different latent features. In detail, the final outputs of the graph neural network are $\mathbf{e}_{u}^{*}$ and $\mathbf{e}_{v}^{*}$, which are the representation for user u and item v respectively. An attention-based rating prediction model can be constructed as follows:

$$\mathbf{a}_{uv} = SOFTMAX\left(\mathbf{W}_{as} \cdot \sigma\left(\mathbf{W}_{a} \cdot \left(\mathbf{e}_{u}^{*} \| \mathbf{e}_{v}^{*}\right)\right)\right) \tag{16}$$

$$\hat{r}_{uv} = \sigma\left(\mathbf{a}_{uv}^{\top} \mathbf{W}_{r} \cdot \left(\mathbf{e}_{u}^{*} \| \mathbf{e}_{v}^{*}\right)\right) \tag{17}$$

where $\hat{r}$ denotes the predicted rating, $\mathbf{a}_{uv}$ measures the importance of each dimension in the latent representation. $\mathbf{W}_{a}, \mathbf{W}_{as}$, and $\mathbf{W}_{r}$ are all learnable parameters. *SOFTMAX* is a normalized exponential function. Please note that GAT is one kind of attention-based graph neural netowrk. It adopts the self-attention strategy to measure the importance of each neighbor node during the state aggregation process. However, in GAT all of the neighbors are attended to learn the target node representation. Thus, its training cost is still high due to neighbor explosion. Compared to GAT, the proposed BIG-SAGE@ adopts an importance sampling-based model to train node representation efficiently, while integrating the attention model to flexibly model the importance of embedding features in the interaction modeling stage.

Finally, we train our model by minimizing the mean squared error loss (MSE) [51], and update these parameters above by the Adam optimizer [52]. The objective function of the model is the following:

$$\mathcal{L} = \min \frac{1}{||\Lambda||} \sum_{(u,v) \in \Lambda} \left(r_{uv} - \hat{r}_{uv}\right)^{2} + \lambda \|\Theta\|_{F}^{2} \tag{18}$$

where Λ represents the training set divided by edges, and $\|\Lambda\|$ is the size of the training set. *Model parameters* $\Theta = \left\{\mathbf{W}_{*}, \widetilde{\mathbf{W}}_{*}\right\}$ are learned by minimizing the objective function (18), $\|\cdot\|_{F}$ is the *Frobenius* norm, λ is the regularization term.

4. Experimental Results

To demonstrate the superiority of BIG-SAGE@ and reveal the reasons for its effectiveness, we conduct extensive experiments on four real-world industrial datasets from Amazon and answer the following research questions(RQs):

- **RQ1**: How does the BIG-SAGE@ perform *w.r.t* rating prediction task as compared with other SOTA models;
- **RQ2**: How are the benefits of performing rating-confidence sampling in graph-based recommendation model;
- **RQ3**: How do different hyper-parameter settings (e.g., embedding dimensions, the adjust factor of rating confidence) influence the effectiveness of proposed BIG-SAGE@,
- **RQ4**:How does the BIG-SAGE@ perform *w.r.t* top-*N* task as compared with other SOTA models.

4.1. Experimental settings

Datasets. We evaluate BIG-SAGE@ on the Amazon datasets with different characteristics, a widely used dataset for product recommendation [53]. We select four sub-category of Amazon [1] including **CD**, **FOOD**, **MOVIE**, and **MUSIC**. From [54], we also use a 10-core setting to ensure the quality of each dataset. The basic statistics of each dataset are described in Table 3.Moreover, we split the dataset chronologically and kept the densest time period. In detail, we select the interactions in the first 80% of time periods in training and the rest in the test. From the training set, we use the last 10% of interactions as a validation set to tune hyper-parameters. Such as in the Amazon Food dataset, we first choose the 10 months period with the densest interactions and retaining the top 5,000 users and items, then we use the ratings from the first 8 months in training and the rest in the test, and we use the last month of interaction data in training set for validation.

Evaluation metrics. In this work, we mainly focus on the MSE and MAE values to evaluate models' performance, which is widely used in the rating prediction tasks [55,56].

$$MSE = \frac{1}{\|\Gamma\|} \sum_{(u,v) \in \Gamma} \left(r_{uv} - \hat{r}_{uv}\right)^{2} \tag{19}$$

The MAE value, namely Mean Absolute Error, is calculated as follows:

[1] http://jmcauley.ucsd.edu/data/amazon

Table 3
The basic statistics of the datasets.

Datasets	#users	#items	#interactions	Density
CD(subgraph)	77,1722	11,402	500,000	0.21%
FOOD	768,438	166,049	1,297,156	0.07%
MOVIE	2,088,619	201,298	4,607,047	0.07%
MUSIC	478,235	266,414	836,006	0.07%

$$MAE = \frac{1}{|\Gamma\|} \sum_{(u,v)\in\Gamma} |r_{uv} - \hat{r}_{uv}| \tag{20}$$

To evaluate the top-N performance of the model, we adopt three widely-used evaluation metrics: Recall@k, which considers whether the relevant items are retrieved within the top-Npositions.

$$Recall@N = \frac{1}{|U|}\sum_{u\in U}\frac{|\widehat{R}(u)\cap R(u)|}{|R(u)|} \tag{21}$$

The NDCG measures the relative orders among positive and negative items in the top-Nlist.

$$NDCG@N = \frac{1}{|U|}\sum_{u\in U}\left(\frac{1}{\sum_{i=1}^{\min(|R(u)|,N)}\frac{1}{\log_2(i+1)}}\sum_{i=1}^{N}\delta(i\in R(u))\frac{1}{\log_2(i+1)}\right) \tag{22}$$

where $\delta()$ is an indicator function. We denote $|U|$ is the size of the user set. $\widehat{R}(u)$ is a list with N recommendation items generated by recommender, while $R(u)$ denotes the item set rated by u in the test set.

Compared models. To evaluate the overall performance, we compared BIG-SAGE@ against the following models:

- **NGCF**[38]: This model introduces the concept of the message to the graph neural network. It uses the weighted average of the initial state of neighbors as the initial message, which is spread layer by layer in the neighbor structure. Then, the messages in each layer are connected in series as the final node representation.
- **PinSAGE**[37]: This model uses random-walk to sample nodes, selects the top N neighbors of the target node according to the L1 norm, and dynamically constructs a new subgraph structure to perform local graph convolution. Moreover, it uses a minibatch for distributed gradient training.
- **neuMF**[10]: This model combines traditional matrix factorization and multilayer perceptrons and can extract low-dimensional and high-dimensional features simultaneously. It first trains the MF model and MLP model separately, and then directly initializes the neuMF model with the trained parameters and performs another round of training.
- **GC-MC**[57]: This model designs a differentiable graph self-encoding framework based on message passing to realize matrix completion while considering side information and network structure.
- **BIG-SAGE**: A graph neural network with rating-confidence-based neighborhood sampling method proposed in this paper, which uses dot product operation to realize rating prediction.

To evaluate the performance of BIG-SAGE@ on top-N task, we compare it with the following top-N recommendation methods:

- **ItemKNN**[58]: It analyzes the user–item matrix to determine the similarities between the different items, and recommends items similar to the previously clicked item.
- **NGCF and neuMF**: We adjusted NGCF and neuMF to achieve top-N recommendation in the same manner as BIG-SAGE@.
- **NGCF-GS**: This is an improved version of NGCF by GraphSaint [45], which constructs minibatches by sampling the training graph. That is, the model is trained on multiple batches of subgraphs, rather than the complete training graph.

Parameter settings: For BIG-SAGE@, we uniformly set the dimension k to 150, and other hyper-parameters all adopt optimal parameters. The range of hyper-parameter k is [50], and the adjustment interval is 20. The range of the hyper-parameter λ is [0.2, 0.8], and the adjustment interval is 0.1. To a fair comparison, the hyper-parameters of baselines are reused according to the respective papers. On the other hand, we tune the hyper-parameters of our proposed model using the validation set and terminate training if validation performance does not improve for 20 epochs, or the related error between consecutive iterations is less than 10^6. All experiments are performed on the same server with the configuration of a GPU with 64 GB memory. The above-mentioned methods are implemented in Python3.

4.2. Performance Comparison with BIG-SAGE@ (RQ.1)

We first evaluate the overall recommendation performance of all mentioned models on different datasets. The comparison results on different datasets are shown in Table 4. From the table, we have the following observations: (1) BIG-SAGE@ achieves the best performance and consistently outperforms all the baselines on the boards. Compared to the strongest baselines on MAE or MSE, BIG-SAGE@ still achieves different degrees of improvement on the five datasets. In detail, BIG-SAGE@ outperforms the strongest baselines in terms of MAE by at least 17.80%, 16.10%, 27.9%, and 7.12% on datasets CD, FOOD, MOVIE, and MUSIC, respectively. It demonstrates the effectiveness of BIG-SAGE@, which is attributed to better capturing collaborative signals in the multiple embedding layers with rating-confidence based neighborhood sampling strategy. (2) Attention mechanism plays a positive role in improving the performance of BIG-SAGE@ on all datasets. Compared to the baseline BIG-SAGE, BIG-SAGE@ adds an attention mechanism to the rating prediction function to weigh the importance of each dimension in the user and item representations. As a result, more important features can participate in rating prediction, while filtering out the noise or irrelevant features.

4.3. Benefits of rating-confidence sampling (RQ.2)

In this part, we conduct several self-comparison experiments with different sampling methods to verify the effectiveness of a rating-confidence based neighborhood sampling method. The purpose of these sampling methods is to sample neighbor nodes for the target node. However, they are based on different distributions, including uniform distribution and distributions based on time information, degree information, and rating confidence. Furthermore, we did not add the attention-based rating prediction. To avoid its potential impact on the experimental results, we adopt the result of the dot product between user and item representa-

Table 4
Over performance on Amazon datasets. We use bold font to denote the best result in each column and use underlined to denote the best baseline result (without BIG-SAGE) in each column. Significance tests are conducted on MSE and MAE between BIG-SAGE@ and the best baseline. $*$ denotes *p-value* < 0.05. *%Improv.* denotes the percentage of improvement on MSE and MAE of BIG-SAGE@ over the best baseline. Lower MSE and MAE mean higher prediction accuracy.

Dataset	CD		FOOD		MOVIE		MUSIC	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
NGCF	<u>1.2109</u>	<u>0.8563</u>	1.5248	0.9591	1.5211	0.9756	<u>0.9199</u>	<u>0.6871</u>
PinSage	1.2367	0.87	1.5296	0.9666	1.3566	0.9404	0.9331	0.7223
neuMF	1.3509	0.9875	1.588	0.971	1.5963	0.9934	0.9476	0.8219
GC-MC	1.3213	0.8923	<u>1.5102</u>	<u>0.9472</u>	<u>1.1305</u>	<u>0.8863</u>	0.9572	0.7091
BIG-SAGE	1.025	0.7658	1.4874	0.9214	0.8452	0.6678	0.9048	0.7147
BIG-SAGE@	**0.9953**	**0.7603**	**1.267**	**0.876**	**0.8152**	**0.6558**	**0.8544**	**0.6774**
%Improv.	17.8%$^{☆}$	11.2%$^{☆}$	16.1%$^{☆}$	7.5%$^{☆}$	27.9%$^{☆}$	26%$^{☆}$	7.12%$^{☆}$	1.4%$^{☆}$

tions as the predicted rating. Fig. 3 shows the MSE and MAE values obtained on the validation set in the BIG-SAGE model using uniform sampling, degree-based sampling, time-based sampling, and rating-confidence-based sampling. Fig. 4 and 5 show the changes in the MSE values on the training set and the test set under different sampling methods.

In summary, we can find the above observations: BIG-SAGE with rating-confidence based sampling achieves the best performance and consistently on four datasets. In detail, rating-confidence achieves a different improvement in rating prediction tasks on all datasets. This verifies the effectiveness of the rating-confidence sampling method. That is because the two important pieces of information (i.e., node degree and created time) are considered in the sampling process, which can obtain more confident neighbor nodes. We notice the the time and node degree play a key role in evulating the edge important in the fired of RS. In RS, the node degree represents the significance or popularity of a node (i.e., user or item node) in the graph, while the time information represent the user's recent perferences on items. Both of them are important to reflect the rating confidence.

4.4. Effectiveness analysis (RQ.3)

BIG-SAGE@ is sensitive to hyper-parameters, i.e., k and λ, making it necessary to conduct parameter sensitivity tests. We present the tuning results concerning k and λ on dataset MOVIE and MUSIC, since highly similar situations can be observed on the other networks for conciseness. First, we uniformly fix λ at 0.7 to validate the performance of BIG-SAGE@ with varied k. Then, we test the impact of λ by setting its value in the range of [0.2, 0.8], while k is uniformly set at 150.

Fig. 6 depicts the tuning process of these parameters on MOVIE and MUSIC, respectively. Table 5 records the optimal parameters on all datasets. From them, we have the following conclusions:

- **The above two hyper-parameters will affect the ability of the test model to converge to the optimal local value.** For example, as shown in Fig. 6a, the minimum value of the objective function appearing at $k = 50$ is 0.8427. The maximum value appearing at $k = 150$ is 0.8152, which is an obvious difference between these two local optima. A similar situation can be seen in other hyper-parameter tests. However, by setting in the range of [0.2, 0.8], the objective function value will not change much, and the difference between the maximum and minimum values is relatively small. With the loss of generality, we can set λ to 0.6 in all datasets.
- **Time information may play a more important role in rating-confidence-based neighborhood sampling.** The known premise is that the larger value λ indicates that the time information is more capable of determining the importance of a target node than the degree information. From Table 5, it can be seen that on most of the data sets, setting $\lambda > 0.5$ may achieve better performance.

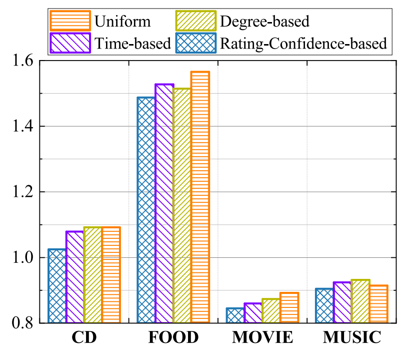
4.5. Performance of BIG-SAGE@ on top-N task (RQ.4)

In this section, we evaluate the performance of BIG-SAGE@ on the top-N task on different datasets. The comparison results on different datasets are shown in Table 6. From the table, we have the following observations:

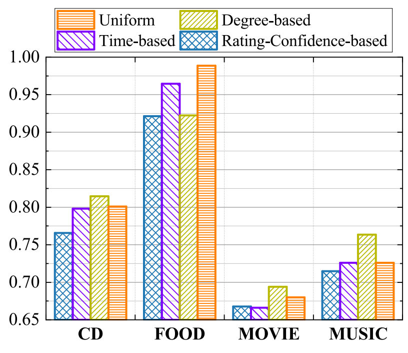
(1) The proposed method achieves the most satisfactory performance. In particular, BIG-SAGE@ outperforms all baselines by an average of 20%, 15%, 9%, and 18% on CD, FOOD, MOVIE, and MUSIC, respectively. It demonstrates that the BIG-SAGE@ also has a superior performance on Top-N task, compared to other baselines. We aslo conducted t-tests, and *p-value* < 0.05 indicates that the improvements of BIG-SAGE@ over baselines are statistically significant.
(2) NGCF generally achieves better performance than neuMF and ItemKNN in most cases. Such improvement might be attributed to the NGCF's capability of exploring the high-order connectivity in an explicit way by stacking multiple embedding propagation layers. In contrast, NeuMF does not explicitly models the connectivity in the embedding learning process, which could easily lead to suboptimal representations. The Item-KNN only utilizes the similarity between items, and cannot capture the high-order connectivity structure of user-item relationships.
(3) For NGCF-GS, as one kind of graph sampling based method, has no improvements over NGCF and BIG-SAGE@ on recommendation performance. The inner reason is that: Graph-Saint adopts a static sampling method to construct subgraph for training, which only fucos on the topological structure. However, time information plays an important role in the recommender system, which can reflects the dynamic change of user preference.

4.6. Discussion and Limitations

Different from the current GNN methods that typically rely on the uniform rule for neighbor node sampling, our BIG-SAGE@ is a more efficient method featuring a rating-confidence neighbor node sampling strategy, providing a reasonable way for dynamic sampling in the real environment. It leverages the rating confidence level to evaluate each neighbor node's importance, combining the time and degree information. Furthermore, an attention mechanism is employed to identify the important features by automatically assigning different weights. Experiment results and ablation studies presented in Section 4.2 and 4.3 demonstrate that our BIG-SAGE@ outperforms other competing methods. To further under-

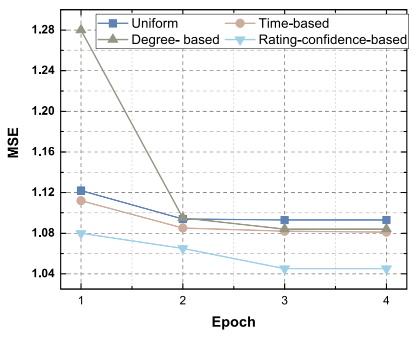


(a) MSE

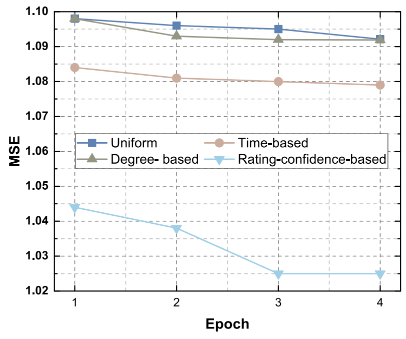


(b) MAE

Fig. 3. BIG-SAGE with different sampling methods for rating prediction.

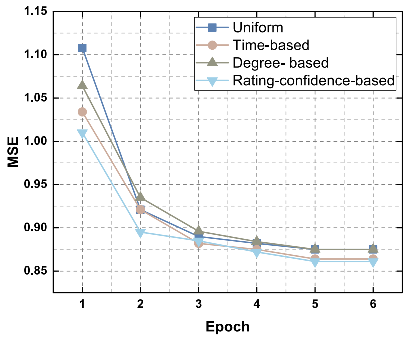


(a) Training Set

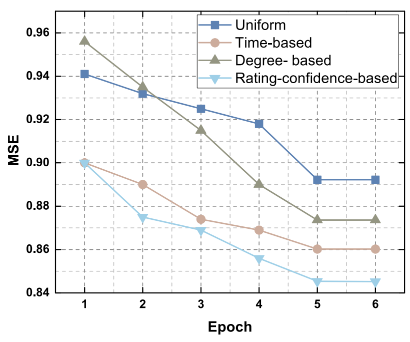


(b) Validation Set

Fig. 4. Convergence Curve of BIG-SAGE@ on dataset CD.



(a) Training Set



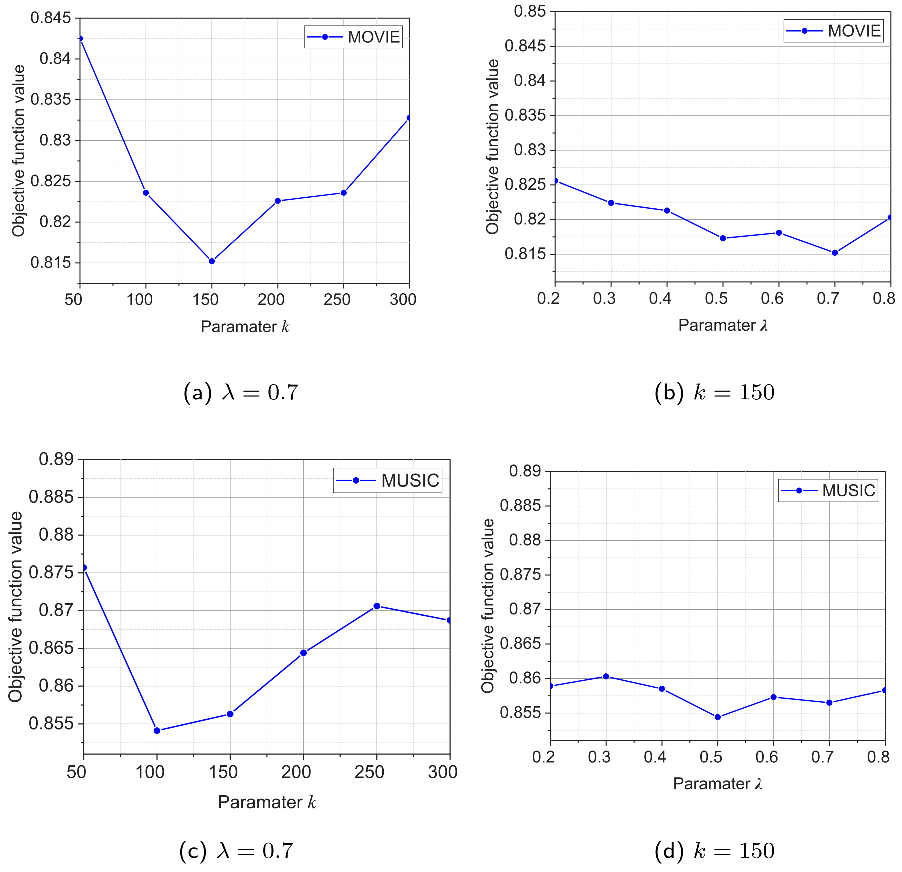
(b) Validation Set

Fig. 5. Convergence Curve of BIG-SAGE@ on dataset MOVIE.

stand the performance of BIG-SAGE@, we carefully analyze the effects of hyperparameters on BIG-SAGE@ (see Section 4.4).

Although our BIG-SAGE@ model achieved good results on the rating prediction task, its performance and practical implementation could be further improved by carefully dealing with the limitations listed as follows:

- Different from the existing sampling method focuses on the unsupervised method, we believe it is a promising way to predict the node importance by leveraging deep neural network. However, how to define the node importance in the graph as the training label is the key to the success of the supervised learning-based methods.
- We sample neighborhood nodes based on timestamps and degree information in this work. However, in fact, time and degree information can also be used as features, which means it can further mine richer node structure features with some distance measurement for measuring node importantce. In addition, how to solve the cold start to provide the new node with neighbor samples for input is also an urgent problem to



(a) $\lambda = 0.7$ (b) $k = 150$

(c) $\lambda = 0.7$ (d) $k = 150$

Fig. 6. The performance of BIG-SAGE@ with different hyper-parameter settings on dataset MOVIE and MUSIC.

Table 5
The optimal hyper-parameters on all datasets.

Parameter	CD	FOOD	MOVIE	MUSIC
k	50	100	150	100
λ	0.6	0.7	0.7	0.5

be solved. Finally, given the low efficiency of dynamic sampling, we can introduce a distributed computing framework. We plan to study these issues in future work.

5. Conclusions

Recently, Graph Neural Network (GNN) has attracted increasing attention and achieved superior performances in various domains (e.g., link prediction, node classification, and recommender system). However, how to dynamically measure the importance of neighbor nodes and embedded features are worthy of discussion in piratical applications. To address this issue, we proposed BIG-SAGE@, a neighbor and feature importance-aware attention rating prediction model, which is applied to the recommender system with a more accurate prediction performance. Specifically, BIG-SAGE@ first introduces the rating-confidence sampling method to identify the important neighbors. Based on the selected neighbor nodes, it leans the node representation via graph neural network with the states aggregation and update process. After that, an attention-based model is employed to construct the rating prediction model by dynamically evaluating each latent feature's importance. Experimental results have been conducted on real-world datasets and demonstrated the superiority of our BIG-SAGE@ in the rating prediction task, compared to several state-of-the-art methods.

Table 6
Performance of BIG-SAGE@ on the top-*10* task on different datasets. We use bold and underline fonts to denote the best performance and second best performance method in each metric respectively. $*$ denotes *p-value*$<$ 0.05. *%Improv.* denotes the percentage of improvement on corresponding metric of BIG-SAGE@ over the best baseline.

Dataset	CD		FOOD		MOVIE		MUSIC	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
ItemKNN	0.0175	0.0098	0.0093	0.0051	0.0461	0.0319	0.0211	0.0127
neuMF	0.0210	0.0110	0.0226	0.0114	0.0610	0.0320	0.0178	0.0096
NGCF	0.0353	0.0181	0.0218	0.0111	0.0699	0.0351	0.0172	0.0086
NGCF-GS	0.0250	0.0129	0.0200	0.0107	0.0620	0.0334	0.0105	0.0085
BIG-SAGE@	**0.0400**	**0.0230**	**0.0250**	**0.0138**	**0.0710**	**0.0410**	**0.0250**	**0.0150**
%Improv.	13.31%$*$	27.07%$*$	10.62%$*$	21.05%$*$	1.57%	16.81%$*$	18.48%$*$	18.11%$*$

CRediT authorship contribution statement

Qingxian Wang: Conceptualization, Methodology, Writing - original draft. **Suqiang Wu:** Data curation, Formal analysis, Software. **Yanan Bai:** Writing - review & editing. **Quanliang Liu:** Software, Validation, Formal analysis. **Xiaoyu Shi:** Supervision, Project administration, Writing - review & editing.

Data availability

The data that has been used is confidential.

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.

Acknowledgments

This research is part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008, HZ2021017), in part by the Chinese Academy of Sciences “Light of West China” Program, in part by Sichuan Science and Technology Program under Grants 2021YFQ0054 and 2022YFG0175, in part by Key Lab of Internet Natural Language Processing of Sichuan Provincial Education Department (No. INLP202201), in part by Key Program of the National Natural Science Foundation of China under Grant No. 82030066, in part by Chongqing Natural Science Foundation (cstc2020jcyj-msxmX0792), and the “Fertilizer Robot” project of Chongqing Committee on Agriculture and Rural Affairs.

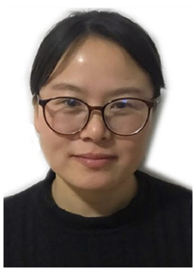
References

[1] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys (CSUR) 52 (1) (2019) 1–38.
[2] P. Resnick, H.R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.
[3] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in artificial intelligence 2009.
[4] 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.
[5] X. Shi, X. Luo, M. Shang, L. Gu, Long-term performance of collaborative filtering based recommenders in temporally evolving systems, Neurocomputing 267 (2017) 635–643.
[6] S. Jalali, M. Hosseini, Collaborative filtering in dynamic networks based on deep auto-encoder, The Journal of Supercomputing 78 (5) (2022) 7410–7427.
[7] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[8] D. Lian, X. Xie, E. Chen, Discrete matrix factorization and extension for fast item recommendation, IEEE Transactions on Knowledge and Data Engineering.
[9] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1235–1244.
[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.
[11] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: A factorization-machine based neural network for ctr prediction, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 1725–1731.
[12] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, G. Sun, xdeepfm: Combining explicit and implicit feature interactions for recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1754–1763.
[13] T.R. Gwadabe, Y. Liu, Improving graph neural network for session-based recommendation system via non-sequential interactions, Neurocomputing 468 (2022) 111–122.
[14] X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1273–1284.
[15] C. Xu, P. Zhao, Y. Liu, V.S. Sheng, J. Xu, F. Zhuang, J. Fang, X. Zhou, Graph contextualized self-attention network for session-based recommendation., in: IJCAI, Vol. 19, 2019, pp. 3940–3946.
[16] Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang, Stamp: short-term attention/memory priority model for session-based recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1831–1839.
[17] H. Liu, C. Zheng, D. Li, Z. Zhang, K. Lin, X. Shen, N.N. Xiong, J. Wang, Multi-perspective social recommendation method with graph representation learning, Neurocomputing 468 (2022) 469–481.
[18] E.R. Bonet, D.M. Nguyen, N. Deligiannis, Temporal collaborative filtering with graph convolutional neural networks, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 4736–4742.
[19] J. Xia, D. Li, H. Gu, T. Lu, P. Zhang, N. Gu, Incremental graph convolutional network for collaborative filtering, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2170–2179.
[20] Y. Koren, Collaborative filtering with temporal dynamics, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 447–456.
[21] X. Zhao, J. Liang, J. Wang, A community detection algorithm based on graph compression for large-scale social networks, Information Sciences 551 (2021) 358–372.
[22] M. Fey, J.E. Lenssen, Fast graph representation learning with pytorch geometric, arXiv preprint arXiv:1903.02428.
[23] H. Peng, B. Du, M. Liu, M. Liu, S. Ji, S. Wang, X. Zhang, L. He, Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning, Information Sciences 578 (2021) 401–416.
[24] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering.
[25] H. Wang, D. Lian, W. Liu, D. Wen, C. Chen, X. Wang, Powerful graph of graphs neural network for structured entity analysis, World Wide Web 25 (2) (2022) 609–629.
[26] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.
[27] X. Li, X. Zhang, P. Wang, Z. Cao, Web services recommendation based on metapath-guided graph attention network, The Journal of Supercomputing (2022) 1–27.
[28] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (1) (2008) 61–80.
[29] T.E. Trueman, P. Narayanasamy, J. Ashok Kumar, A graph-based method for ranking of cloud service providers, The Journal of Supercomputing 78 (5) (2022) 7260–7277.
[30] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[31] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, Computer Science.
[32] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Lió, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[33] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, P.S. Yu, A survey on heterogeneous graph embedding: methods, techniques, applications and sources, arXiv preprint arXiv:2011.14867.
[34] H. Wang, D. Lian, Y. Zhang, L. Qin, X. He, Y. Lin, X. Lin, Binarized graph neural network, World Wide Web 24 (3) (2021) 825–848.
[35] C. Li, K. Jia, D. Shen, C.-J.R. Shi, H. Yang, Hierarchical representation learning for bipartite graphs., in: IJCAI, Vol. 19, 2019, pp. 2873–2879.
[36] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, X. He, Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2347–2357.
[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, 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, 2018, pp. 974–983.
[38] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019, pp. 165–174.
[39] M. Schutera, M. Hussein, J. Abhau, R. Mikut, M. Reischl, Night-to-day: Online image-to-image translation for object detection within autonomous driving by night, IEEE Transactions on Intelligent Vehicles 6 (3) (2020) 480–489.
[40] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, F. Nashashibi, Attention based vehicle trajectory prediction, IEEE Transactions on Intelligent Vehicles 6 (1) (2020) 175–185.
[41] A. Luo, P. Zhao, Y. Liu, F. Zhuang, D. Wang, J. Xu, J. Fang, V.S. Sheng, Collaborative self-attention network for session-based recommendation, IJCAI (2020) 2591–2597.
[42] J. Wang, K. Ding, Z. Zhu, J. Caverlee, Session-based recommendation with hypergraph attention networks, in: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM) SIAM, 2021, pp. 82–90.
[43] D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, Q. Gu, Layer-dependent importance sampling for training deep and large graph convolutional networks, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS, 2019.
[44] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, 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.
[45] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, GraphSAINT: Graph sampling based inductive learning method, in: International Conference on Learning Representations, 2020.
[46] J. Chen, J. Zhu, L. Song, Stochastic training of graph convolutional networks with variance reduction, in: International Conference on Machine Learning, PMLR, 2018, pp. 942–950.
[47] J. Chen, T. Ma, C. Xiao, Fastgcn: Fast learning with graph convolu-tional networks via importance sampling, in: International Conference on Learning Representations, International Conference on Learning Representations ICLR, 2018.
[48] Z. Chen, L. Li, J. Bruna, Supervised community detection with line graph neural networks, in: International Conference on Learning Representations, 2019.
[49] W. Chen, F. Feng, Q. Wang, X. He, C. Song, G. Ling, Y. Zhang, Catgcn: Graph convolutional networks with categorical node features, IEEE Transactions on Knowledge and Data Engineering.
[50] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in neural information processing systems 31.
[51] D. Wu, M. Shang, X. Luo, Z. Wang, An l1-and-l2-norm-oriented latent factor model for recommender systems, IEEE Transactions on Neural Networks and Learning Systems doi:10.1109/TNNLS.2021.3071392.
[52] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
[53] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: Proceedings of the 25th international conference on world wide web, 2016, pp. 507–517.
[54] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 639–648.
[55] D. Wu, Q. He, X. Luo, M. Shang, Y. He, G. Wang, A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction, IEEE Transactions on Services Computing.
[56] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, X. Wu, A data-characteristic-aware latent factor model for web services qos prediction, IEEE Transactions on Knowledge and Data Engineering.
[57] R. van den Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, 2018.
[58] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 143–177.

Qing-Xian Wang received her Ph.D. degree in computer science from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006. She joined the School of Computer Science and Engineering, UESTC, Chengdu, China, in 2002, then worked in the School of Information and Software Engineering in 2015, and is now an associate professor of Information and Software Engineering at UESTC. Her research interests are in recommender systems, artificial intelligence, automatic reasoning, and information security.

Suqiang Wu received the BE degree in University of Electronic Science and Technology of China (UESTC) in 2020. He is currently studying for a MS degree at UESTC. His research interests include deep learning and recommender systems.



Yanan Bai received the M.S. degree in computer science from the Guizhou University, Guizhou, China, in 2010, and the PhD degree in computer science from the University of Chinese Academy of Sciences, Chongqing, China, in 2022. She joined the Chongqing University of Technology, Chongqing, China, in 2022. Her research interests are in information safety and artificial intelligence.

Quanliang Liu received the BE degree in software engineering from the Jiangxi University of Science and Technology, Jiangxi, China, in 2021. He is currently a master degree candidate in computer science from the University of Chinese Academy of Sciences, Chongqing, China, in 2021. His research interests are in recommender system and reinforcement learning.

Xiaoyu Shi (Member, IEEEE) received the BE degree in computer science from Information Engineering University, Zhengzhou, China, in 2007, and the PhD degree in computer science from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2015.He joined the Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), Chongqing, China in 2015, and is currently an associate professor of computer science and engineering. His research interests include machine learning, recommender systems.