ECCF: An Improved GNN Collaborative Filtering Algorithm for Edge Computing

Weifeng Sun (wfsun@dlut.edu.cn)
Dalian University of Technology

Kangkang Chang
Dalian University of Technology

Bowei Zhang
Dalian University of Technology

Research Article

Keywords: GNN, Edge computing, Over-smoothing, Calculate lightweight

Posted Date: February 2nd, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2532581/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.
ECCF: An Improved GNN Collaborative Filtering Algorithm for Edge Computing

Weifeng Sun¹*, Kangkang Chang¹ and Bowei Zhang¹

¹School of Software, Dalian University of Technology, Dalian, Tuqiang Street, Dalian, 116620, Liaoning, China.

*Corresponding author(s). E-mail(s): wfsun@dlut.edu.cn; Contributing authors: ckk19970825@mail.dlut.edu.cn; beta@mail.dlut.edu.cn;

Abstract

There are many relational data in edge computing. They belong to non-Euclidean spatial data. Graphical neural network can expand edge computing capabilities to extract features from non-Euclidean spatial data, and thus is suitable for edge computing data classification and filtering. In IoT edge computing, there are many computing nodes and data to be transmitted. Therefore, how to transmit the most needed data to the computing nodes is an important issue, and a better recommendation model is needed to filter the edge data. LightGCN is a superior recommendation model based on graph neural network, but there is still a problem of over-smoothing when the user-item relationship extraction layer is stacked with multiple layers. To solve this problem, this paper proposes an improved graph neural network collaborative filtering algorithm (ECCF) to build a deeper graph neural network to achieve better feature extraction, keep the purity of local features, and improve recommendation performance. ECCF uses hard negative mining method as a sampling method to improve recommendation accuracy and training efficiency, and uses IndRNN with long and short-term memory with self-attention layer to capture valid information. The experimental results of the model using Gowalla, Yelp 2018 and Amazon Book show that ECCF performs better in NDCG and Recall evaluation metrics than LightGCN, IMP-GCN and NGCF. ECCF improves the recommendation accuracy while minimizing the complexity of the model, which is well adapted to be applied to data filtering for edge computing.

Keywords: GNN, Edge computing, Over-smoothing, Calculate lightweight
1 Introduction

With the development of the Internet of Things (IoT), marginal computing can process data from the edges generated by more data, which meets the needs of more delayed sensitive applications, and has quickly become the focus of the current research. Applications in edge computing often need to serve a large number of users, which means that edge servers need to handle large amounts of data. However, the capacity of edge servers to handle the data is limited due to their limited computing capacity, which will potentially cause higher latency and reduced quality of service. We can preliminarily filter the transmitted data to reasonably select the effective data to improve the quality of service. Applying the recommended algorithm to edge computing is an important means of data filtering \[1, 19, 20\]. In order to perfectly deal with the data generated by edge devices, Cao et al. \[1\] integrate the time information and geographic information of user check-in in LBSN, and propose a POI recommendation algorithm that comprehensively considers edge devices and cloud, which has high recommendation accuracy and can address the cold start problem. Yu et al. \[19\] propose a recommendation algorithm based on predictive QoS and user geographic location (MDITCF) to recommend appropriate services to users. The algorithm combines collaborative filtering and user geographic location, and comprehensively considers the characteristics of edge services, mobility and user needs in different time periods to solve the cold start problem. They also introduce the idea of inverse CF Rec into traditional CF, predicted the lost quality of service (QoS), and solve the problem of sparse data. Zhang et al. \[20\] analyze the communication process of locator/identity separation network, and propose and implement a cooperative filtering based fixed identity mapping prediction algorithm (FIMPA) to solve the problem of delay in locator/identity separation protocol (LISP) mapping resolution and cache update.

The primary research goal of recommendation algorithms applied to edge computing is still to improve the accuracy of general recommendation. The recommendation models based on graph neural network perform well on data with a large number of user-user and user-item relationships, and is suitable for edge computing data filtering. Among the many GCN-based models, Light-GCN proposed by He et al. \[4\] learns user and item embeddings by linearly propagating them over the user-item interaction graph, and uses the weighted sum of embeddings learned at all layers as the final embedding, making the model easier to train. However, if we want to further improve the performance of the LightGCN model and build a deeper image convolution layer, the model will become difficult to train. Moreover. The deeper layers also propagate noisy information from the extended neighborhood, i.e., there is an over-smoothing problem of the graph neural network.

Although many researchers try to solve the problem of over-smoothing, for example, using residual connection \[6\], the problem still exists. In our previous paper \[15\], we used multi-layer IndRNN to alleviate the problem of over-smoothing. Based on this, further experiments related to the influence of layers
are conducted and deeply analyzed, and ECCF is proposed. ECCF is improved on the basis of LightGCN. The over-smoothing problem is attempted to be solved by using a 4-layer IndRNN [8] after the user item vector is performed with Concatenate operation.

The main contributions of this paper are as follows:

1) Based on LightGCN, the use of IndRNN that can maintain long-term and short-term memory can effectively capture neighborhood information, while maintaining the purity of local features to alleviate the problem of over-smoothing.

2) A 4-layer IndRNN is identified as the superior layer, which not only significantly improves the model performance, but also saves a lot of training time to adapt to the edge computing environment.

3) We use three open datasets to calculate the NDCG, Recall and sparsity effects of ECCF, and compare them with LightGCN, NGCF [17] and IMP-GCN [9]. The results show that ECCF recommendation is better. The results of ablation experiments also show the effectiveness of our improved part.

The remaining parts of this article are as follows. In section 2, we introduce the research work related to recommendation algorithm based on graph neural network. Section 3 is mainly about our improvement of LightGCN and explains the structure of ECCF and the role of each part. Section 4 introduces the experimental environment, experimental data and analysis of experimental results. Finally, we summarize this article in Section 5.

2 Related work

Some deep learning algorithms [7, 10, 18] are well integrated with edge computing scenarios and take up important recommendation tasks. In order to accurately capture the transformation mode between POIs in edge computing, while protecting the location of users, Kuang et al. [7] propose a model of user sign in sequence and their potential states based on HMM, and use EM algorithm to estimate the parameters of the model. They also protect the user’s location information by weighted noise injection method, and based on Forward algorithm predict the user’s next move according to the user’s current location. Wu et al. [18] propose a group recommendation system that uses knowledge graph and LSTM in edge computing for web document resource mining. The system can effectively solve the problem of information overloading and resource trekking. It has conducted extensive system tests in the big data application field of the packaging industry. The system uses knowledge graph and LSTM in edge computing to recommend network document resources more accurately, which can meet the user’s personalized resource requirements. In order to solve the problem of sensitive information leakage in the recommendation system, Liang et al. [10] integrat multi-access edge computing networks into the recommendation system to make full use of base stations and user terminals. They extract user profiles and recommendation algorithms from the cloud server and sent them to the base station and user.
ECCF: An Improved GNN Collaborative Filtering Algorithm for Edge Computing

terminals. Considering the differences between user terminals, they also propose a general matrix decomposition framework, which can adopt different matrix decomposition-based recommendation algorithms (IFMF).

Graph neural network has many advantages. It is good at processing non-Euclidean spatial data with irregular structure, has the ability to handle various relationships of data, and can efficiently use the structural characteristics among sample instances. In general, graph neural network-based data recommendation algorithms can be divided into two categories, general recommendation and serialized recommendation. General recommendation aims to find out long-term stable interests of users, and use user-item interaction information to model user preferences. In recent years, NGCF, LR-GCN [2], LightGCN, and IMP-GCN are the better general recommendations. NGCF learns user and item embeddings through linear propagation on the user-item interaction graph, and uses the weighted sum of embeddings learned in all layers as the final embedding, which has achieved good experimental results. Later, some scholars try to solve the problem of over-smoothing of GCNs. LR-GCN removes the nonlinearity in GCN, simplify the network structure, and introduces residual network structure to alleviate over-smoothing. Light-GCN considers that some feature transformations and nonlinear activation in NGCF have no positive impact, and removes them from the model to alleviate over-smoothing. IMP-GCN considers that multi-layer graph convolution will enable users with different interests to have similar embeddedness. It uses subgraphs composed of users with similar interests and their interaction terms for training, so that the model can avoid propagating negative information from high-order neighborhoods to embedding learning.

Unlike general recommendation, serialized recommendation aims to accurately predict the next item that users may be interested in. To make sequential recommendation, we need to extract as much valid information as possible from the sequence to understand users’ short-term interests, long-term interests, etc. The sequence recommendation representatives based on graph neural network include GES-SASRec [21], RetaGNN [5], SGRec [12], MA-GNN [13], STP-UDGAT [11] and GPR [3]. RetaGNN has the ability of induction and transferability, and the learnable weight matrix in the model is based on various relationships among users, items and attributes, which is more conducive to prediction and recommendation. GES-SASRec enhances the target item representation by combining the upper and lower items of the target item in different sequences. SGRec makes full use of the POI representation of cooperative signal learning, and fully reveals multi-level sequential pattern user preference modeling to overcome the sparsity of POI-level interaction. MA-GNN applies graph neural networks to model item contextual information in the short-term cycle, and captures long-term dependencies between items using shared memory networks. Lim et al. proposes a spatiotemporal preference user dimension graph attention network (STP-UDGAT), which utilizes personalized user preferences and new POI in STP neighborhood, while allowing users to selectively
learn from other users. In addition, STP-UDGAT considers geographical location, timestamp, frequency and other factors in the recommendation. The GPR model uses potential geography influenced by trained inputs and outputs to represent user interests.

In order to further improve the accuracy of the general recommendation model and make the model better assume its role in edge computing, we use the structure of IndRNN to alleviate the problem of excessive smoothing and try to find a better model structure. The next part introduces the model structure.

3 ECCF

3.1 Architecture

![Overall structure of ECCF](image)

The overall structure of the model is shown in Figure 1. ECCF performs graph convolution iteratively and learns the representation of nodes through the features on the graph. The main components of the model are: propagation embedding layers, feature processing layers, feature extraction enhancement layers and score calculation. The propagation embedding layers is used to obtain the high-order connection relationship of nodes. The feature processing layers mainly performs the concatenate operation on the previous relational vectors. The feature extraction enhancement layer is used to solve the problem of excessive smoothness, and obtains data feature relationships other
than connection relationships, and then calculates the score. The user vector is represented as $e_u$, and the item vector is represented as $e_i$. Next, this paper introduces the content details of propagation embedding layer, feature extraction enhancement layer and scoring calculation in detail.

### 3.1.1 Propagation embedding layers

This layer uses graph convolution to aggregate connected neighbor nodes to obtain the connection relationship of nodes. Stacking multi-layer propagation embedded layer can obtain the high-order connection relationship information of each node, which is very important for calculating the correlation between user and item. In order to improve the recommendation performance and reduce the difficulty of model training, ECCF uses the same graph convolution as LightGCN to remove the feature transformations and nonlinear activation functions that are useless for improving the recommendation effect, and the expression is as formula (1):

$$
e^{(k+1)}_u = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|} \sqrt{|N_i|}} e^{(k)}_i,$$

$$
e^{(k+1)}_i = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|} \sqrt{|N_u|}} e^{(k)}_u. \tag{1}$$

Where, $e^{(k)}_u$ and $e^{(k)}_i$ represent the embedding expression of user (u) and item (i) after k-layer propagation respectively, $N_u$ represents the interactive item set for user (u), and $N_i$ represents the interactive user set for item (i). $\frac{1}{\sqrt{|N_i|} \sqrt{|N_u|}}$ is a symmetric normalization term, which can avoid the increase of embedding scale with graph convolution operation, and has good performance.

In the propagation embedding layer, we only aggregate connected neighbors rather than self-connection. This is because layer combination operations basically achieve the same results as self-connection. The embedding expression obtained after layer k of the model is as formula (2):

$$
e_u = \sum_{k=0}^{K} \alpha_k e^{(k)}_u; \quad e_i = \sum_{k=0}^{K} \alpha_k e^{(k)}_i, \tag{2}$$

where, $\alpha_k$ represents the weight of layer k embedding in the final embedding. There are three reasons why we execute layer combination to obtain the final representation. 1) As the number of layers increases, the embedding will be over-smoothed. 2) Different layers of embeddings capture different semantics. For example, the first floor is forced and smooth for users and projects with interaction, the second layer smooths out users (items) with overlap on items (users), and higher layers capture higher-order proximity. Therefore, combining them will make the presentation more comprehensive. 3) Combining embedding at different levels with weighted sum can capture the graph convolution effect with self-connection.
3.1.2 Feature extraction enhancement layers

In order to further obtain data feature relationship beyond the user-item connection relationship, we design the feature extraction enhancement layers. After the Concatenate operation, we add four layers of IndRNN layer and one layer of self-attention mechanism\[16\] layer.

After Concatenate operation, enter $e^*_t$ to enter IndRNN. The status update of IndRNN can be described as formula (3):

$$h_t = \text{ReLU} \left( W e_t + u \odot h_{t-1} + b \right),$$  \hspace{1cm} (3)

where $e_t$ is the hidden state and input vector of time step $t$, $W$ is the input weight, and $u$ is the recursive weight. $\odot$ is the product of Adama, and $b$ is the offset value. The activation function is selected as ReLU. Each layer of neurons in IndRNN is independent, and the connection between neurons is realized by adding more layers. For the $n$-th layer, hidden state $h_{n,t}$ can be obtained by formula (4):

$$h_{n,t} = \text{ReLU} \left( W_n e_t + u_n h_{n,t-1} + b_n \right),$$  \hspace{1cm} (4)

where, $W_n$ and $u_n$ are the input weights and recursive weights of the $n$-th layer respectively. Each neuron only receives information from the input and its own hidden state in the previous step. The IndRNN infrastructure is shown in Figure 2.

![Fig. 2 Infrastructure of IndRNN](image)

We input the output results of the IndRNN layer into the self-attention mechanism layer. The self-attention mechanism can reduce the dependence on external information and is good at capturing the internal correlation and linear characteristics of data or features. The expression of self-attention mechanism is shown in formula (5):

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{Q K^T}{\sqrt{d_k}} \right) V,$$  \hspace{1cm} (5)
where Q represents the Query vector, K represents the Key, and V represents the Value. \( d_k \) represents the dimension of each head of the self attention mechanism, while \( \sqrt{d_k} \) is the scale factor, which is used to avoid excessive inner product value and help the training converge more smoothly. The algorithm uses the same feature extraction method for vector \( e_i^* \).

The basic calculations of the self-attention mechanism are all matrix calculation, which can solve the length-dependence problem of the sequence, obtain the explicit characteristics of the data, and perform parallel calculation. It is widely used in image vision and recommendation system. In this paper, we use the self-attention mechanism to extract features in the last step to enhance the representation ability of feature vectors. The output \( e_u' \) and \( e_i' \) are two vectors that have fully extracted features.

### 3.1.3 Score calculation

This layer calculates the prediction score. We use the inner product of the final expression of user and item as the score, as shown in formula (6):\[
\hat{y}_{ui} = e_u'^T e_i'
\] (6)

### 3.2 Explanation of IndRNN

The reason why IndRNN can alleviate the problems of gradient disappearance, explosion and over-smoothing: for the gradient backpropagation through time in each layer, the gradient of IndRNN can be calculated independently for each neuron, because there is no interaction between them in a layer. For the \( n \)-th neuron \( h_{n,t} = \sigma(W_n e_t + u_n h_{n,t-1}) \) (where \( \sigma \) represents the ReLU activation function), it is assumed that \( J_n \) is the target of minimization in time step \( T \) when the deviation is ignored. The gradient of backpropagation to time step \( t \) is formula (7):

\[
\begin{align*}
\frac{\partial J_n}{\partial h_{n,t}} &= \frac{\partial J_n}{\partial h_{n,T}} \frac{\partial h_{n,T}}{\partial h_{n,t}} = \frac{\partial J_n}{\partial h_{n,T}} \prod_{k=t}^{T-1} \frac{\partial h_{n,k+1}}{\partial h_{n,k}} \\
&= \frac{\partial J_n}{\partial h_{n,T}} \prod_{k=t}^{T-1} \sigma'_{n,k+1} u_n = \frac{\partial J_n}{\partial h_{n,T}} u_n^{T-t} \prod_{k=t}^{T-1} \sigma'_{n,k+1},
\end{align*}
\] (7)

where \( \sigma'_{n,k+1} \) is the derivative of the element activation function. It can be seen that the gradient only involves the exponential term of the easily adjustable scalar value \( u_n \) and the gradient of the activation function, which is usually bounded within a certain range. Compared with the gradient \( \left( \frac{\partial L}{\partial h_{T}} \prod_{k=t}^{T-1} \text{diag} (\sigma' (h_{k+1})) U^T \right) \) of RNN, the gradient of IndRNN directly depends on the value of recursive weight, rather than the matrix product. Therefore, we can solve the problem of gradient explosion and disappearance...
ECCF: An Improved GNN Collaborative Filtering Algorithm for Edge Computing

over time by adjusting the exponential term $u_n^{T-t} \prod_{k=t}^{T-1} \sigma'_{n,k+1}$ to an appropriate range. In the following, we need to calculate the appropriate range of gradient.

By assuming that the minimum effective gradient is $Q$, the recursive weight range of IndRNN neurons can be obtained to maintain long-term memory. According to formula (7), in order to maintain the T-t time step, $|u_n| \in \left[ (T-t) \sqrt{\frac{\epsilon}{\prod_{k=t}^{T-1} \sigma'_{n,k+1}}}, +\infty \right)$. The gradient disappearance of neurons can be avoided only if the above constraints are met. We further limit the range to $|u_n| \in \left[ \sqrt{(T-t)} \sqrt{\frac{\epsilon}{\prod_{k=t}^{T-1} \sigma'_{n,k+1}}}, (T-t) \sqrt{\frac{\gamma}{\prod_{k=t}^{T-1} \sigma'_{n,k+1}}} \right]$, where $\gamma$ is the maximum gradient without explosion. For commonly used activation functions, for example, ReLU and tanh, their derivatives are not greater than 1, i.e., $|\sigma'_{n,k+1}| \leq 1$. Better short-term memory is also important for network performance, especially for multi-layer RNN, the constraint on recursive weight range with ReLU activation function can be relaxed to $|u_n| \in [0, (T-t) \sqrt{\gamma}]$. When the recursive weight $u_n$ is 0, the neuron only uses the information from the current input, and does not retain any memory from the past. With the constraints of the above parameters, different neurons can learn to keep memories of different lengths.

3.3 Hard negative mining

To improve the accuracy of the model, we choose hard negative mining\cite{14} as the sampling method. Hard negative samples refer to samples with relatively large loss, which means that it is easy to regard negative samples as positive samples.

Steps of hard negative mining:

1) In target detection, positive and negative training sets are divided according to labeled data. With tagged terms, the terms with the intersection and combination ratio (IoU) greater than the set threshold are positive samples, otherwise they are negative samples. Generally, negative samples are far more than positive samples. In order to avoid that the trained model tends to predict negative samples, it is necessary to maintain sample balance. The initial negative sample training set needs to select a subset of the negative sample set.

2) Train the neural network. After training the model, use the trained model to predict the remaining negative samples. According to the set threshold, if the predicted positive probability is greater than this threshold value, the term can be added to the negative sample training set.

3) The positive sample training set unchanged, while the negative sample training set adds the new data obtained in step 2) in addition to the initial data. Use the new training sets to start a new round of training.
3.4 Model optimization method

The ECCF model uses the BPR loss function, as shown in formula (8):

$$\text{Loss} = \sum_{(u,i,j) \in O} -\ln \sigma (\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2,$$  

(8)

where $O = \{(u, i, j) \mid (u, i) \in R^+, (u, j) \in R^-\}$ represents paired training data, $R^+$ represents observed interactions, and $R^-$ represents unobserved interactions. $\sigma$ is a sigmoid function; $\Theta = \{E, \{W_1^{(l)}, W_2^{(l)}\}_{l=1}^L\}$ represents all trainable model parameters. $\lambda$ is a coefficient, which controls the intensity of L2 adjustment to prevent over fitting. We use Adam to optimize the prediction model and update the model parameters.

4 Experiments and results

4.1 Experimental settings

**Experimental environment**: Tensorflow 1.4 framework, Python 3.6, RTX3080 GPU, and Windows 10 operating system are used. The embedded size of the model is 64, the batch size is set to 1024, and the learning rate is set to 0.001.

**Datasets**: We use Gowalla, Yelp 2018 and Amazon Book. These three data sets can represent data in edge computing.

Gowalla: Data set is a record of 1679245 social friend relationships and visited poi among 329839 users in Gowalla from September 2011 to December 2011, as well as the type classification of visited places.

Yelp 2018: Data set is taken from the Yelp Challenge in 2018. It contains the name, address and ID of the business. Restaurants and bars are considered as projects.

Amazon Book: Amazon bestseller data set. It contains the top 50 best sellers per hour. It contains time, book ranking, book name, user evaluation and other information.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>users</th>
<th>Items</th>
<th>Interactions</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla</td>
<td>29858</td>
<td>40981</td>
<td>1027370</td>
<td>0.00084</td>
</tr>
<tr>
<td>Yelp2018</td>
<td>31831</td>
<td>40841</td>
<td>1666869</td>
<td>0.00128</td>
</tr>
<tr>
<td>Amazon-Book</td>
<td>52643</td>
<td>91599</td>
<td>2984108</td>
<td>0.00062</td>
</tr>
</tbody>
</table>

**Baselines**: The selected baselines are LightGCN, IMP-GCN and NGCF. Since the performance of INGCFCF [6] is lower than IMP-GCN as a whole, this article does not take it as a baseline. LightGCN is our main comparison model,
while IMP-GCN and NGCF are excellent graph neural network recommendation algorithms in recent years. The comparison results with these models can better reflect our improvement effect.

LightGCN: This model learns user and project embeddedness by linear propagation of user and project embeddedness on the user project interaction graph, and uses the weighted sum of embeddedness learned at all levels as the final embeddedness. This simple, linear and concise model is easier to implement and train. LightGCN with three graph convolutional layers has good performance, so we use it for comparison.

IMP-GCN: This model designs an unsupervised subgraph generation module, which can effectively identify users with common interests by using user characteristics and graphic structures. The IMP-GCN of four sub-charts always performs very well, so we use it as the baseline.

NGCF: This model is used to express and model the higher-order connectivity in the user project diagram, so as to effectively inject the collaboration signal into the embedding process in an explicit way. LightGCN also has good performance of the three graph convolutional layers, so we use it for comparison.

4.2 Evaluation indicators

In order to evaluate ECCF, we compare the Recall and NDCG values with other algorithms. Recall is the proportion of positive classes in the sample that are predicted correctly. The expression is shown in formula (9):

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

TP represents that the original positive class is predicted to be positive, and FN represents that the original positive class is predicted to be negative.

DCG and NDCG are usually used to measure and evaluate algorithms related to recommendation and sorting. The calculation method of DCG is shown in formula (10):

$$\text{DCG} = \text{rel}_1 + \sum_{i=2}^{pm} \frac{\text{rel}_i}{\log_2 i} \quad (10)$$

As shown above in (10), pm represents the quantity of all documents returned in this query. rel$_i$ indicates the relevance of the $i$-th returned document to this query. According to the traditional method, the relevance is generally set as 1, 2 and 3 respectively. The larger the number, the greater the correlation. Because the same conditions must be set for pm when DCG is used, which is not conducive to calculation and comparison, this paper uses NDCG as the evaluation index instead of DCG. IDCG represents the DCG value when pm setting is optimal. The NDCG expression is shown in Formula
\[ \text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \] (11)

4.3 Research on the number of layers

In order to study the impact of the number of IndRNN layers on the overall model, we use the method of increasing the IndRNN layer from 1 to 7, calculating the Recall and NDCG values of the ECCF model for comparison, so as to determine the optimal number of IndRNN layers. The experimental results are shown in Figure 3.

![Fig. 3 Impact of IndRNN layers on results](image_url)

From the above results, we can conclude that with the increase of the IndRNN layer, the Recall and NDCG values of the ECCF model have always increased. This shows that stacking IndRNN can improve the performance of the model and effectively alleviate the problem of over-smoothing. However, when the IndRNN layer increases from the first layer to the fourth layer, the performance is greatly improved. From the fifth layer, the performance improvement becomes smaller and tends to converge. The time required to train 5-layer and above IndRNN models has been greatly increased. For example, under the experimental environment of RTX3080 GPU, the training time of 5-layer model is about 0.5 times longer than that of 4-layer model. It
is unnecessary to spend so much time, so we choose the four-layer IndRNN ECCF model.

### 4.4 Performance comparison

#### Table 2 Overall performance comparison

<table>
<thead>
<tr>
<th></th>
<th>Gowalla</th>
<th>Yelp-2018</th>
<th>Amazon-Book</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall@20</td>
<td>NDCG@20</td>
<td>Recall@20</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.1553</td>
<td>0.1332</td>
<td>0.0438</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.1796</td>
<td>0.1529</td>
<td>0.0649</td>
</tr>
<tr>
<td>IMP-GCN</td>
<td>0.1869</td>
<td>0.1585</td>
<td>0.0667</td>
</tr>
<tr>
<td>ECCF</td>
<td>0.1924</td>
<td>0.1671</td>
<td>0.0686</td>
</tr>
<tr>
<td>%Improv.</td>
<td>2.94</td>
<td>5.42</td>
<td>2.84</td>
</tr>
</tbody>
</table>

We still use Gowalla, Yelp2018 and Amazon Book data sets for experiments and calculation models Recall@20 and NDCG@20 to compare the performance of each model. The experimental results are shown in Table 2.

According to Table 2, we can conclude that:

1) The performance of LightGCN is better than NGCF generally. LightGCN uses a simple weighting and aggregator, while removing feature transformation matrices and nonlinear activation functions that are useless for recommendations. It reduces the operation cost and improves the recommendation performance.

2) IMP-GCN has better performance than LightGCN. This is because IMP-GCN groups users and their interaction items into different subgraphs, and performs higher-order graph convolution in the subgraphs. By using user characteristics and graph structure to effectively identify users with common interests, the model can avoid spreading the negative information of high-order neighbors to embedded learning. IMP-GCN can alleviate the problem of over-smoothing to some extent.

3) Compared with the above baseline, ECCF achieves the best performance on all data sets. ECCF has better performance than IMP-GCN. Recall@20 In Gowalla, Yelp 2018 and Amazon Book, the growth rate is 2.94%, 2.84% and 2.83% respectively; NDCG@20 In Gowalla, Yelp 2018 and Amazon Book, they increase by 5.42%, 5.17% and 5.28% respectively. This is because ECCF effectively alleviates the problem of gradient explosion and over-smoothing, and can achieve better recommendation results by stacking more layers, and use hard negative mining to improve the accuracy of the model.

### 4.5 Performance comparison at different sparsity levels

We have carried out experimental research on the important sparsity of recommendation algorithm. We select user groups with different sparsity levels from Gowalla, Yelp2018 and Amazon Book datasets for experiments. For each data set, we select 5000 users. Based on the interaction time of each user, we
divide the test sets into 8 groups. The interaction times of each user in these 8 groups are less than 25, 50, 100, 200, 400, 600, 800, 1000 respectively. Figure 4 shows the comparison results.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{comparison_results.png}
\caption{Sparsity experimental structure of three datasets}
\end{figure}

According to Figure 4, the experimental results of the ECCF model are optimal on three data sets with different sparsity. In particular, the 600 and 800 groups of Yelp 2018 exceed the best baseline by 3.94% and 3.78%, respectively. These data show that data sparsity has little impact on ECCF, and ECCF model can solve the problem of data sparsity to some extent.

4.6 Ablation experiments

This part will reflect the impact of feature extraction layer and hard negative mining on the overall performance of the model. The results are shown in Table 3. In Table 3, FU-ECCF represents the complete model of ECCF, NE-ECCF represents the model of ECCF without feature extraction enhancement layer, and NH-ECCF represents the model of ECCF using traditional sampling mechanism.

It can be seen from the data that without using the feature extraction enhancement layer, Recall@20 decrease by 2.3% and NDCG@20 decrease by
6.23% in the Gowalla dataset; Recall@20 decrease by 6.29% and NDCG@20 decrease by 3.53% in the Yelp2018 dataset; dataset Recall@20 decrease by 8.73% and NDCG@20 decrease by 6.70% in the Amazon Book. These data fully illustrate the advantages of feature extraction enhancement layer in improving the effect of feature extraction and mitigating over smoothing. Comparing the values of NH-ECCF and FU-ECCF, it is found that hard negative mining also improves the accuracy of the model.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Results of ablation experiments by using three data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LightGCN</td>
</tr>
<tr>
<td>Gowalla</td>
<td>Recall@20</td>
</tr>
<tr>
<td></td>
<td>NDCG@20</td>
</tr>
<tr>
<td>Yelp2018</td>
<td>Recall@20</td>
</tr>
<tr>
<td></td>
<td>NDCG@20</td>
</tr>
<tr>
<td>Amazon-Book</td>
<td>Recall@20</td>
</tr>
<tr>
<td></td>
<td>NDCG@20</td>
</tr>
</tbody>
</table>

4.7 Analysis of experimental results

ECCF increases Recall by at least 2.83% and NDCG by at least 5.17% on the three data sets. Compared with three excellent models, the overall recommendation performance of ECCF model can be improved. This is because the feature extraction enhancement layer of ECCF model can effectively capture neighborhood information, maintain the purity of local features, and alleviate the problem of excessive smoothing. The experimental results of the number of layers show that the four-layer IndRNN can greatly improve the accuracy of model recommendation without significantly increasing the complexity of the model. The results of ablation experiments show that the effectiveness of our improved part and the model structure are better. The ECCF model improves the recommendation accuracy without significantly increasing the complexity, so it is very suitable for data filtering related tasks in edge computing.

5 Conclusions and future work

This paper proposes an improved graph neural network recommendation algorithm ECCF based on LightGCN for edge computing. This algorithm retains the advantage of LightGCN’s strong ability to capture higher-order connection features of graph data, and adds four layers of IndRNN layer and feature extraction enhancement layer constructed by self-attention mechanism layer after the Concatenate operation of user-item vectors at all levels of LightGCN. The combination of IndRNN and self-attention mechanism in the model makes the model more robust and generalizable, and alleviates the problem of over-smoothing after multiple layers of graph neural network are stacked to a certain
extent. Our team change the sampling method to improve the model discrimination performance. The ECCF model structure is suitable for data filtering related tasks in edge computing. We explore the optimal number of IndRNN layers and conduct data sparsity impact experiments. This paper compares NDCG and Recall indicators of ECCF, IMP-GCN, LightGCN and NGCF on three public data sets, and conducts ablation experiments to illustrate the effectiveness of our improvements in improving recommendation performance.

In the future work, we will try to use more data sets to evaluate the performance of ECCF, and consider the recommendation effect when applied to edge computing with more data in the network.

Acknowledgments. This work is supported by National Key R&D Program of China (2018YFB1700100), CERNET Innovation Project (NGII20190801) and the Fundamental Research Funds for the Central Universities under Grants (DUT21LAB115).

References


in edge computing. Security and Communication Networks **2020** (2020)


[21] Zhu, T., Sun, L., Chen, G.: Graph-based embedding smoothing for sequential recommendation. IEEE Transactions on Knowledge and Data Engineering (2021)