Sun *et al. Journal of Cloud Computing* (2023) 12:110
https://doi.org/10.1186/s13677-023-00480-0

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Recommend what to cache: a simple self-supervised graph-based recommendation framework for edge caching networks

Aijing Sun[1], Guoqing Wang[1*] and Qi Han[2]

Abstract

Deep Learning-based edge caching networks can accurately infer what to cache based on a user's historical content requests, thereby significantly relieving the burden of the backbone networks. However, the cold-start problem inherent in deep learning may limit the performance of history-based caching strategies. Due to the mobile and dynamic nature of wireless networks, base stations often lack sufficient data to accurately estimate the user's demands and cache the possible requested data. In this context, we adopt self-supervised learning (SSL) into the caching strategies and propose a Simple Self-supervised Graph-based Recommendation framework for edge caching networks (SimSGR). Specifically, we propose two new network layers: the Mixing layer and the Conversion layer. The former replaces the data augmentation of the SSL paradigm to avoid destroying the semantic loss, while the latter greatly simplifies the loss function, which helps to lighten the model structure and facilitates deployment on edge caching networks. Simulation results show that our model outperforms baseline algorithms that are sensitive to augmentation hyper-parameters, particularly when trained in a cold-start environment.

Keywords Edge caching, Cold-start, Self-supervised, Graph-based recommendation

Introduction

The proliferation of modern mobile devices has led to a plethora of applications, such as VR video, mobile gaming, and so on [1]. Such applications demand low latency and high bandwidth from data stores, usually hosted in cloud data centers. However, the large transmission latency incurred by the long distance between mobile users and the cloud data center may significantly degrade the quality of service on mobile devices [2]. To mitigate the impact of distance, Mobile Edge Computing (MEC) has been proposed and deployed to move contents' proximity to the network edge and proactively cache popular contents [3]. Compared with traditional centralized cloud computing, MEC can greatly reduce access latency and load on backhaul, core, and transit networks [4]. From the perspective of content caching, wireless edge caching can greatly improve the efficiency of content delivery by deploying storage and caching facilities at the network edge [5].

A closely related problem to be solved by edge caching is content placement, that is, determining what, where, and when to cache [6]. Several cache strategies, including Least Recently Used (LRU), Least Frequently Used (LFU), and First In First Out (FIFO), were proposed successively. Recently, with the continuous increase of data size and types, many researchers have proposed to introduce machine learning into the edge network [7]. These learning strategies could accurately predict the demand for data content by tracking and leveraging the user's history demand to recommend the network what to cache. However, the recommendation

*Correspondence:
Guoqing Wang
wgq_rainbow@163.com
[1] School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[2] ShenZhen ZET Technology & Service Company Limited, Shenzhen, China

system usually suffers from the cold-start problem in the edge caching network. When a new user's mobile devices enter a new cell, the base stations have no sufficient prior knowledge of the new user [6]. In this way, it cannot obtain an accurate estimation of the demand and cache the possible requested data.

Recently, a resurgence of Self-supervised learning (SSL) has been witnessed in the recommendation system [8]. In the SSL task, the supervision signals could be automatically generated from the raw data instead of human-provided labels, which is a natural antidote to the cold-start problem in the Edge caching network based on deep learning [9]. More precisely, the existing graph-based SSL recommendation (e.g., Self-supervised graph learning [10]) typically performs stochastic augmentation, such as random node/edge dropout, to perturb the raw user-item bipartite graph to generate different views. Then the augmented versions are fed into a shared encoder to learn representations. The contrastive learning (CL) task, the core paradigm of the SSL task, aims to maximize the consistency of positives and push the semantics of negatives apart. Significantly, the representations of the same node (user/item) but learned from different views are regarded as positives. On the contrary, the remaining nodes in one batch are regarded as negatives. Finally, the main task of recommendation and the CL task will be jointly training to build effective representations.

Although SSL methods are effective, we argue that it still has the following defects:

1) *Arbitrary augmentations aggravate the cold-start problem:* The SSL paradigm based on augmentation was first proposed in the computer vision domain. As shown in Fig. 1 (a), an input image is augmented to generate different views. Despite random cropping and rotating them or disturbing their color, the image's inherent semantics can still be well preserved, and humans can recognize the changes [11]. Unlike data augmentation well defined in images, the augmentations may behave arbitrarily on graphs. For example, in Fig. 1 (b), randomly dropping nodes or edges may change the semantics of graphs. More specifically, whether Bob or the edge between Bob and Movie1 is dropped, Movie1 will not be recommended to Ruby. Moreover, the relationship between Mike and Ruby will become distant, such random node/edge dropping will separate the original graph into many disconnected components and destroy the mutual information between the augmented graphs [9]. Therefore, the number of informative positives that can be used for SSL will be reduced, thus aggravating the cold-start problem.
2) *The computationally expensive learning strategy:* In the classic SSL, the CL task usually plays the role of regularization terms and are jointly trained with the main task of the recommendation system. Although the joint learning strategy is effective to some extent, training for two tasks simultaneously is too time-consuming and memory-cost, especially for the edge caching network with limited computing resources [12]. Considering that the downstream task in most recommendation scenarios is fixed, that is, predicting the possible behavior of users. here we propose a new learning strategy in SimSGR

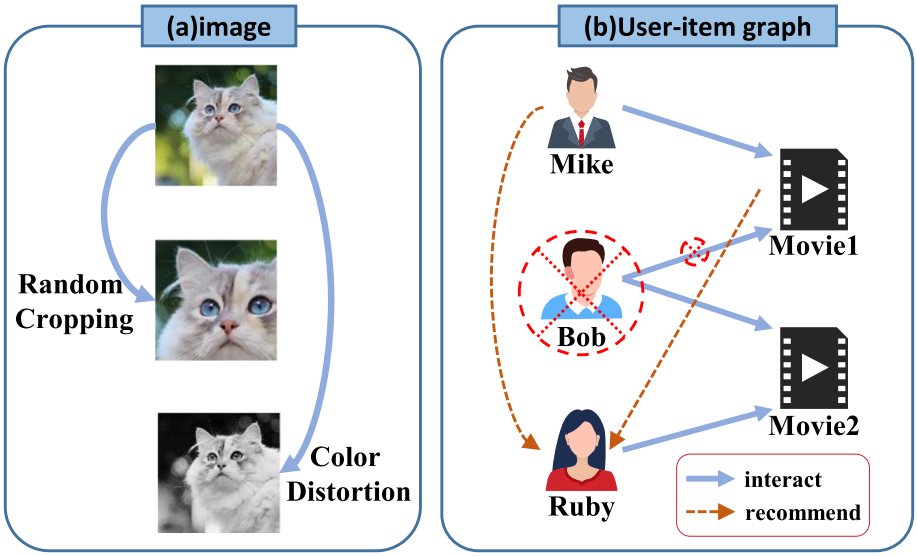

Fig. 1 **a** Augmentations on images preserve the inherent semantics. **b** Augmentations behave arbitrarily on user-item graphs

which could unify the CL task and the main the main task of the recommendation.

To solve the preceding limitations, we propose a Simple Self-supervised Graph-based Recommendation framework for edge caching network (SimSGR), where the core is to modify the generation of positives based on two layers: Mixing and Conversion.

A) *Mixing:* We still follow the paradigm of CL task in the SSL, which is to maintain the invariance of the representation by pulling in positives from two views. However, positives are no longer generated by randomly perturbing the topology of the original graph. Instead, we use the original graph as one view and mix another view, both of which no longer need augmentations, thus avoiding the semantic change of the graph caused by augmentation.
B) *Conversion:* Unlike the existing CL, which directly pulls the positives together in the representation space, we add the Conversion layer before calculating the loss function. In this layer, the predicted rating matrix is calculated based on the representation. Thus, our positives are updated to the corresponding elements in two matrices from two views. In this way, the positives are transformed from the representation space to the rating space, which is consistent with the main task of the recommendation system. This simple step unifies the CL task and the main task of the recommendation system.

The major contributions of this paper are summarized as follows:

(1) We introduce SSL into edge caching networks to alleviate the cold-start problem of caching strategies.
(2) We further tailor a new SSL framework for edge caching networks and propose a new model SimSGR, based on the Mixing layer and the Conversion layer. This model can generate highly reliable positives and simplify the model structure, thus saving the computing resources of the edge network while improving the model performance.
(3) Extensive experiments demonstrate that our SimSGR outperforms a wide range of state-of-art methods, especially in the cold-start scenario.

Related work

SimSGR proposes two new network layers Mixing and Conversion to reconstruct the positive pairs. The former is designed to replace the data augmentation on the graph, and the latter is used to modify the criterion of CL. Therefore, we first introduce the existing augmentation methods on the graph in SSL, then discuss the different designs of the loss function in CL and conclude some typical SSL-based recommendations.

Augmentations on graphs

While intuitive augmentations are well defined on images and texts, they may behave arbitrarily on graphs due to their discrete, non-euclidean nature [13] As a consequence, augmentations for graphs are less explored. DGI [14] first proposes to augment the input graph via shuffling node features. MVGRL [15] adopt the graph diffusion kernels to generate two augmented views of the original graph. GCC [16] proposes to create multi-views of a graph via sampling subgraphs. GRACE [17] randomly drops edges and masks features to augment the input. Following GRACE, GraphCL [18] designs four graph augmentations types by injecting different priors and demonstrates the effects of various combinations of data augmentations on different datasets. Different from the methods mentioned above adopting random data augmentation schemes to construct positives, GCA [19] designs a new augmentation scheme according to the node centrality to corporate more priors for topological and semantic information of the graph.

However, due to the natural complexity of graphs, the mentioned SSL-based model highly depends on the data augmentation scheme [11], and [18] has shown that there is no universally outperforming data augmentation scheme for graphs. AFGRL [11] first proposes to generate positive pairs based on the local structural information and the global semantics of graphs instead of data augmentation, where the structure of the original graph will not be destroyed, thus preserving the semantics. Simultaneously, SimGCL [9] discards the augmentation and creates contrastive positives by adding uniform noises to the embedding space. Our SimSGR is mainly inspired by AFGRL, but we modify the details according to the characteristics of the recommendation and remove the clustering strategy in AFGRL, which is redundant for our model.

Loss function in contrastive learning

The main principle of designing the loss function in the CL task is to keep the invariance between the two augmented views while avoiding the model's collapse. A typical loss function InfoNCE aims to pull positive samples together and push negative samples apart. In this way, different definitions of negative samples appear in various methods. SimCLR [20] regards all other samples in one batch as negatives, while MOCO [21] requires an additional memory bank to store negative samples. While

these methods yield good performance, they require a lot of extra storage space.

Instead of avoiding the model's collapse by pushing away the negatives, BYOL [22] has indicated that collapse can be avoided by using architectural tricks without negative sampling. Barlow Twins [11] proposes a loss term to disentangle the embedding variables to maximize the information stored by the embeddings. VICReg [23] introduces two regularization terms, variance and covariance, successfully avoiding dimension collapse.

Inspired by Barlow Twins and VICReg, we introduce covariance regularization to avoid collapse. More importantly, we design a new layer Conversion where the contrastive positives are converted from embedding space into rating space, thus, unifying the CL task and the main task of the recommendation system. In this way, there is no need for negative sampling or joint learning.

SSL-based recommendation

Recently, a wave of researchers attempted to introduce SSL into the recommendation system to alleviate the issue of data sparsity in the recommendation system and have achieved promising results. Sˆ 3-Rec [24] randomly masks the item features to create sequence augmentations. Inspired by SimCLR [20], SGL [10] introduces the InfoNCE loss function into recommendations and augments the original user-item bipartite graph via node/edge dropout and random walk. SEPT [25] attempts to use semi-supervised learning to generate more positives for the social recommendation. CL4Rec [26] reorders and crops item sequences to augment the input for sequential data augmentation. CoSeRec [27] proposes two informative augmentations named 'substitute' and 'insert' to create more informative positives according to the item correlations. A sea of studies have proved that introducing SSL into the recommendation system is conducive to improving performance and solving the existing issues in the recommendation system. However, the mentioned methods all create contrastive positives by data augmentation, which will cause semantic loss. Furthermore, the SSL task is often regarded as an auxiliary task and is jointly trained with the main task of the recommendation, which is both time-consuming and memory-cost. To solve the above problems, we propose a new SSL scheme tailored to the characteristics of graph-based recommendation by reconstructing the contrastive positives. The Details will be given in chapter 3.

Methods

In this section, by reconstructing positives based on Mixing and Conversion, we propose a simple but effective self-supervised graph-based recommendation framework without data augmentation, joint-learning, or even negative sampling. We will elaborate on how to modify the definition of positives and the benefits of this definition. The overall Framework of SimSGR is shown in Fig. 2.

Preliminaries

In the graph-based recommendation system, let $G = (V, \varepsilon)$ denote a graph, where the node set $V = U \cup I$ involves all users and items, and the edge set $\varepsilon \in O^+$ represents the history interactions. G is associated with a feature matrix $X \in \mathbb{R}^{(n+m)\times d}$ and an adjacency matrix $A \in \mathbb{R}^{(n+m)\times(n+m)}$ where $A_{i,j} = 1$ if $(v_i, v_j) \in \varepsilon$ and $A_{i,j} = 0$ otherwise, n is the number of users, m is the number of items and d is the embedding size.

Unlike the existing models where the representations of users (items) are encoded from the augmented graphs, we directly encode the original graph without any augmentation. Specifically, the adjacency matrix A and the feature matrix X of the original graph are fed into the encoder to compute the representation. Here we use the state-of-art graph-based encoder LightGCN [28]:

$$Z = LightGCN(X, A), \tag{1}$$

whose i-th rows, z_i is the representation of node i. We use the representation Z encoded by the original graph as one view and then mix another view.

Mixing layer

In this section, we focus on how to generate the representation of the other view Z' based on the representation Z. It mainly consists of two steps: selecting positives and mixing positives.

Selecting positives

Considering the inherent relational inductive bias in the graph structure, for each target node $i \in V$, its adjacent nodes tend to have the same semantics as the target node. Consequently, we first randomly samples K_{pos} nodes from the neighboring nodes to construct the initial positive samples A_i.

To further validate the effectiveness of A_i, we conduct the following experiments on the Cora dataset. For each node i in the Cora dataset, we calculate the proportion r_i of nodes in A_i that belong to the same class as the target node i. The average of all r_i constitute the final result, where the hyper-parameters K_{pos} is set to $\{4, 8, 16, 32, 64\}$. As shown in Fig. 3, only 70% of the nodes in A_i belong to the same class as the target node, which indicates the unfiltered A_i contains noise nodes of about 30%, namely "false" positives that are semantically contrary to the target node and will lead to poor performance when regarded as positives.

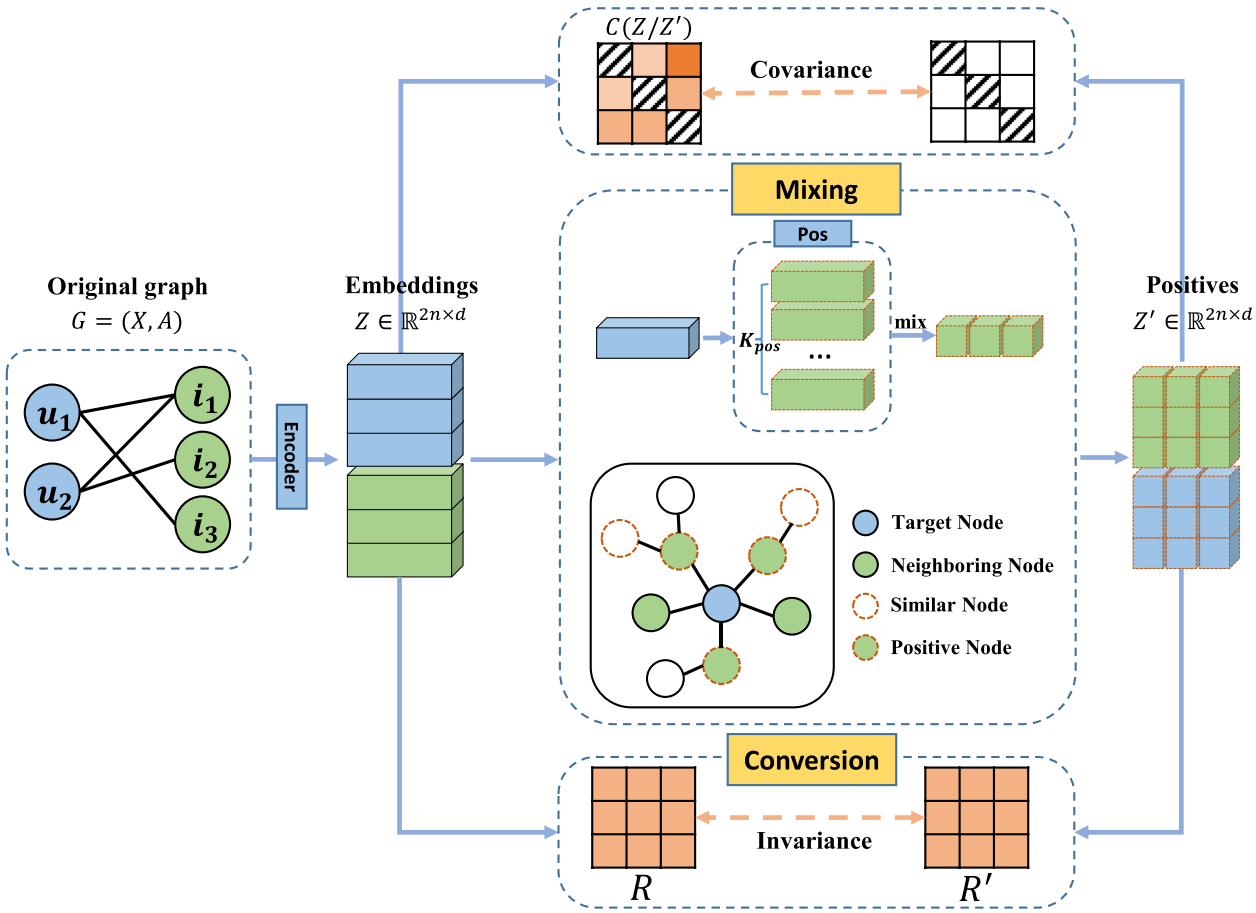

Fig. 2 The overall framework of SimSGR. Following the scheme of CL, one view of the contrastive positive pair Z is encoded from the original graph G, and the other Z' is generated from the Mixing layer by mixing both the neighboring nodes and similar nodes. The two contrastive views are then converted into the rating matrices R and R' in the Conversion layer, SimSGR aims to maintain the invariance of R and R', and the covariance criterion is used to prevent the model from collapsing

Therefore, to filter out "false" positives in the neighboring nodes, following [11], we then compute the cosine similarity between all other nodes in the graph as follows:

$$sim(v_i, v_j) = \frac{z_i \cdot z_j}{\|z_i\| \|z_j\|}, \forall v_i \in V. \tag{2}$$

Given the similarity information, we denote the top K_{pos} nodes with the greatest similarity in the neighboring nodes by the final positive sample set P_i. For each node i, when the number of its neighboring nodes $A_i < K_{pos}$, we expand the size of the positives equal to K_{pos} on padding. The Fig. 3 shows that the ratio of P_i is much higher than that of A_i, which is basically stable at around 90%. This indicates that screening the initial positive samples based on similarity can help the Mixing layer filter out noise and improve the performance of the model.

Mixing positives

Given the positive sample set P_i, the core lies in the aggregation layer for mixing positive representations. The aggregated representation can be obtained as follows:

$$z'_i = \sum\nolimits_{k \in P_i} I_k \cdot \alpha_k z_k, \tag{3}$$

where z'_i is the final representation serves as the positive of the target node i, $z_k \in \mathbb{R}^{d \times 1}$ is the d-dimensional representation of nodes in the positive sample set P_i, α_k denote the weight of the aggregation, I_k denotes the mask indices during padding, where $I_k = 0$ indicates a padding token; otherwise $I_k = 1$. The Calculation of α_k will change with different aggregation methods. Here we adopt three common aggregation strategies, including average-pooling, self-attention, and target-attention [29]:

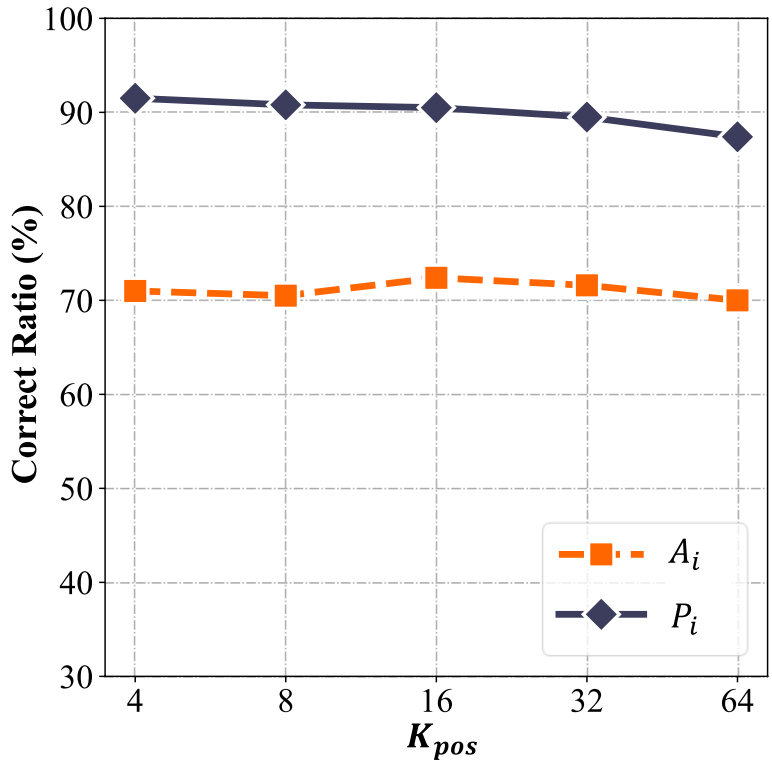

Fig. 3 The ratio of its neighboring nodes being the same label as the target node across different K_{pos}

$$\alpha_k = \begin{cases} \frac{I_k}{\sum_{k \in p_i} I_k} & \text{average - pooling,} \\ \frac{I_k \cdot \exp(q^T \tanh(W_1 z_k + b_1))}{\sum_{j \in p_i} I_k \cdot \exp(q^T \tanh(W_1 z_j + b_1))} & \text{self - attention,} \\ \frac{I_k \cdot \exp(z_i^T \tanh(W_2 \times z_k + b_2))}{\sum_{j \in p_i} I_k \cdot \exp(z_i^T \tanh(W_2 \times z_j + b_2))} & \text{target - attention,} \end{cases} \tag{4}$$

where average pooling treats all alike to average all node representations in P_i. Although it is simple and direct, it does not consider the relative importance of different nodes in P_i and the relationship with the target nodes z_i. According to the attention mechanism, $q \in \mathbb{R}^{d \times 1}$ is a learnable query vector for self-attention and $z_i \in \mathbb{R}^{d \times 1}$ is the target-specific query vector for target node i in target-attention. $W_1, W_2 \in \mathbb{R}^{d \times d}$ and $b_1, b_2 \in \mathbb{R}^{d \times 1}$ are learnable parameters.

The above three aggregation methods are widely used in the embedding aggregation related to deep learning, and their performance also changes with different networks and datasets.

Conversion layer

After aggregation, we finally get a pair of positive representations $Z = [z_{u1}, ..., z_{un}] | Z = [z_{i1}, ..., z_{in}]$ and $Z' = [z'_{u1}, ..., z'_{un}] | Z' = [z'_{i1}, ..., z'_{in}]$. During self-supervised training, most existing loss functions aim to push the positive pair of nodes close to each other in representation space. However, in the main task of the recommendation, our goal is to make the score predicted from the model based on the historical interaction between users and items consistent with the real score. The commonly used loss functions include cross-entropy loss, mean squared error loss, and Bayesian Personalized Ranking (BPR) loss. Taking BPR loss as an example, which aims to improve the accuracy of the recommendation model by maximizing the score difference between rated and unrated items:

$$L_{BPR} = -\sum_{(u,i,j) \in D} \ln \sigma(r_{ui} - r_{uj}) + \lambda \|\Theta\|^2, \tag{5}$$

where D is the training set, (u, i, j) is a triplet where u represents the user, i represents the rated item, and j represents the unrated item. r_{ui} represents the predicted score of user u for item i, while r_{uj} represents the predicted score of user u for item j. σ is the sigmoid function and λ is the regularization term that prevents the model from overfitting. Finally, θ represents all model parameters. The optimization direction of BPR is to pull the predicted score and the real score together in the rating space, which is different from the optimization direction of SSL task. Therefore, existing SSL-based recommendations usually jointly optimize the main task of

recommendation and SSL task. Considering the complexity of joint learning, we propose a new learning strategy which could unify the SSL task and the main task of the recommendation.

We first separate Z and Z' into $Z = Z_u \| Z_i$ and $Z' = Z'_u \| Z'_i$ according to the index of users and items, where || represents the separation operation. We then calculate the predictive rating matrix:

$$R = Z_u (Z_i)^T, R' = Z'_u (Z'_i)^T, \tag{6}$$

where $R \in \mathbb{R}^{n \times n}$,$R' \in \mathbb{R}^{n \times n}$, n is the size of one batch. Following the core idea of pulling two positive samples together in CL, the loss function can be given by [30]:

$$s(R, R') = \frac{1}{n \times n} \sum_k \sum_j \left\| r_{k,j} - r'_{k,j} \right\|^2, \tag{7}$$

where $s(R, R')$ named invariance encourages the scores predicted by two views close to each other, $r_{k,j}$ and $r'_{k,j}$ represent the elements of row k and column j in matrix R and R' respectively.

Furthermore, if we only consider pulling in positives, the model is easy to collapse, which means that all embeddings shrink to a constant value [13]. To prevent collapse with all the input converge to a trivial solution, we first calculate the covariance matrix $C(Z)$ as

$$C(Z) = (Z - \overline{Z})^T (Z - \overline{Z}), \tag{8}$$

where $\overline{Z}$ is a normalized matrix. Inspired by Barlow Twins [13] and VICReg [23], whose core idea is to disentangle the representation and reduce the correlation between dimensions, so that prevent them from holding the same information. Therefore, we define the covariance criterion c as:

$$c(Z) = \frac{1}{d} \sum_{k \neq j} [C(Z)]^2_{k,j}, \tag{9}$$

this term named covariance calculate the sum of the squared off-diagonal coefficients of $C(Z)$ and encourages the off-diagonal coefficients of $C(Z)$ equals to zero.

Based on (7, 8, 9), we propose a new IC (Invariance Covariance) loss that consists of the weighted sum of invariance and covariance:

$$L(Z, Z') = s(R, R') + \lambda [c(Z) + c(Z')], \tag{10}$$

where λ is a hyper-parameter indicating the weight of the covariance criterion.

Complexity

In this section, we will analyze the time complexity of SimSGR and compare it with that of LightGCN and its graph-augmentation based counterpart, SGL. We will focus on batch time complexity since the in-batch negative sampling is a widely used technique contrastive learning. Let $|E|$ be the edge number in the graph, d represent the embedding size, B denote the batch size, M represent the number of nodes in a batch, and ρ denote the edge keep rate in SGL. We can derive: (Table 1)

For LightGCN and SimSGR, no graph augmentations are required. Therefore, they only need to normalize the original adjacency matrix, which has $2|E|$ non-zero elements. On the contrary, SGL requires two graph augmentations, each has $2\rho|E|$ non-zero elements. In the graph convolution stage, SGL employes a three-encoder architecture to learn augmented node representations. As a result, the time cost of SGL is almost three times that of LightGCN and SimSGR. In terms of recommendation loss, both LightGCN and SGL require joint-learning strategies, resulting in an additional complexity $O(2Bd)$ compared to SimSGR. When calculating the CL loss, SGL incurs computation costs of $O(Bd)$ and $O(BMd)$ for the positive/negative samples, respectively, because each node only considers itself as the positive, while the other nodes all are negatives. In contrast, by computing the covariance matrix $C(Z) \in R^{d \times d}$, negative sampling is avoided in SimSGR, resulting in a lower complexity of $O(Bd + Bd^2)$.

Comparing SimSGR with SGL, it is clear that SimSGR theoretically spends less time on all components. This is mainly due to the Mixing and Conversion layers proposed in this paper. These layers greatly simplify the framework of self-supervised recommendation networks, making them more suitable for edge caching networks with resource limitations.

The overall framework

In summary, SimSGR is derived from the traditional SSL paradigm and tailored for the characteristics of edge caching networks. The core of SimSGR is to reconstruct positives based on two layers: Mixing and Conversion. Specifically, in the Mixing layer, we discover each node's most similar nodes based on graph structure and semantic similarity, and then mix them as another view of this node. In this setting, data augmentation is no longer needed, preserving the semantics in the original graph and generating more reliable positives to fundamentally

Table 1 The comparison of time complexity

Component	LightGCN	SGL	SimSGR
Adjacency Matrix	$O(2\|E\|)$	$O(2\|E\| + 4\rho\|E\|)$	$O(2\|E\|)$
Graph Convolution	$O(2\|E\|Ld)$	$O((2 + 4\rho)\|E\|Ld)$	$O(2\|E\|Ld)$
BPR Loss	$O(2Bd)$	$O(2Bd)$	-
CL Loss	-	$O(Bd + BMd)$	$O(Bd + Bd^2)$

solve the cold-start problem. In addition, in the Conversion layer, unlike other methods that treat the representation from two views as positives, we directly calculate the prediction score matrix under the two views. The CL task changes to maximize the consistency of the two matrices, unifying the CL task and the main task of the recommendation. Thus, SimSGR can achieve model performance comparable to joint-learning even when only training on self-supervised tasks. Finally, we disentangle the representation and reduce the correlation between dimensions, ensuring that the trained representations do not collapse even without negative sampling. These improvements simplify SimSGR's structure compared to traditional deep learning model, making it more suitable for edge caching networks with limited resources.

Results and discussion

Experiment settings

Datasets

We evaluate the performance of SimSGR on three popular benchmark datasets, including Douban-book, MovieLens-1 M and Yelp2018, for which the statistics are shown in Table 2. Following the convention in [25] to convert the explicit ratings with a 1–5 rating scale in Douban-book into implicit ratings, we leave out the ratings less than 4 and set the rest at 1. MovieLens-1 M is a popular movie dataset where each record represents the user's rating of the film. We use the processed version downloaded from the Recbole library [30]. Moreover, Yelp2018 was collected from Yelp.com, a subset of the businesses, reviews, and user data. To keep a comparison fair, we directly use the same '10-core' version as [28]. These datasets are split into three parts (training set, validation set, and test set) with a ratio of 8:1:1. Two standard metrics: Recall@K and NDCG@K are used and we set $K = 20$.

Baselines

We consider the following representative baseline methods of three groups and their variants to compare with SimSGR:

a) *Supervised models:* NGCF [31] first proposed the GNN-based recommendation; LightGCN [28], which discards feature transformation and nonlinear activation in NGCF, achieves better results in lighter structures and has become the state-of-art baseline.
b) *Self-supervised models with augmentations:* SGL [10] is a typical model that adopts SSL to the recommendation. Here we use its three variants which represent three different ways of data augmentation: node dropout (SGL-ND), edge dropout (SGL-ND), and random walk (SGL-RW).
c) *Self-supervised models without augmentations:* Unlike group b, all methods in group c generate positives by perturbing the output of the backbone instead of augmenting the input graph. SelfCF [32] first proposed three perturb methods: historical embedding perturbation (SelfCF-HE), embedding dropout (SelfCf-ED), edge pruning (SelfCF-EP); AFGRL [11] discovers positives by considering the local structural information and the global semantics of graphs; Moreover, to regulate the uniformity of the representation distribution, SimGCL [9] adds random noises to the representation to generate positives for contrast.

Table 2 Statistics of the experimented data

Dataset	User	Item	Interaction	Density
Douban-book	13024	22347	792062	0.0027
MovieLens-1M	6040	3952	1000209	0.0419
Yelp-2018	31668	38048	1561406	0.0013

Implementation details

We use the code provided by the author for SGL, SelfCF, AFGRL, and SimGCL and implement NGCF and LightGCN based on the Recbole library [30]. For a fair comparison, we directly refer to the best hyper-parameter settings recorded in the original papers of the baselines and then fine-tune all the hypermeters with the grid search. The values for each hyper-parameter are shown in Table 3. We adopt early-stopping if the performance does not improve within ten rounds in the validation set and record the results on the test set.

Overall performance

In this section, we evaluate and compare our SimSGR with the other baselines mentioned in chapter 4.1.2 on 3 real-world datasets. The hyper-parameters K_{pos}, λ, μ are set to [8, 1, 0.1] on MovieLens-1 M and [4, 1, 0.1] on Douban-Book and Yelp2018. We bold the best performance

Table 3 Values of hyper-parameters

Hyper-parameter	Value
Embedding size	64
Batch size	4096
Optimizer	Adam
Learning rate	0.001
K_{pos}	{4, 8, 16, 32}
λ	[0.1, 0.9]

and underline the second-best. Besides, the improvements are calculated based on LightGCN. According to Table 4, we can draw the following conclusions:

1) Our SimSGR outperforms all baselines with a vast improvement, proving that the framework designed in our paper is conducive to improving the performance of the recommendation. The improvement is mainly due to the reconstruction of the positives: 1) Mixing instead of Augmenting, which will not destroy the original graph's semantics; 2) We convert the loss criteria from the representation space to the rating space, which leads to the unification of the SSL task and the main task of the recommendation.
2) Almost all SSL-based models outperform supervised models, especially on sparse datasets, which indicates that the recommendation system benefits from SSL strategies. When on the recall, SimSGR can remarkably improve LightGCN by 36.78% in Douban-Book, 8.00% in MovieLens-1 M, and 16.30% in Yelp2018. The SSL task helps the recommendation expand the range of input data and provide more positives and negatives for learning representations invariant to the interference factors, thus increasing the model's generality.
3) Self-supervised models without augmentations can achieve or even exceed the performance of self-supervised models with augmentations. SelfCF-HE, AFGRL, and our model SimSGR outperform SGL in most cases, demonstrating that generating credible positive pair is the main driving force of the performance improvement, while graph augmentations are just a way to produce positives, which can be replaced or removed.

The impact of each structure of SimSGR

The core of SimSGR is to reconstruct positives based on two layers, Mixing and Conversion. To verify the advantages of each component of SimSGR, we conduct ablation studies on three datasets.

The performance of mixing layer

In the mixing layer, for each target node, we construct the initial positive set based on its neighboring nodes and then filter out "false" positives according to representation similarity to generate the final positive set, which is recorded as K_P. Here we calculate two degenerate positive sets K_N and K_S for further studies, where K_N represents the positive set based only on neighboring nodes, and K_S represents the positive set based only on the node similarity in the representation space. Next, we combine the preceding three positive sets with the three aggregation methods mentioned in Sect. "Mixing Layer" in pairs and Fig. 4. shows the performance after the combination of three datasets on average-pooling (Avg), self-attention (Self), and target-attention (Target). The performance improves with color deepening. We observe that:

1) Compared with combinations containing K_N and K_S, the combination containing K_P achieves the best performance of all three datasets, which proves that considering both the inductive bias of the graph and the similarity of representation is beneficial for generating positives with high reliability, thus leading to better performance;
2) In the combination containing K_P, Avg achieves the best performance compared with the other two aggregations. However, when the positive set become K_N or K_S, there is no obvious advantage among the three aggregations, which indicates that the aggregation method needs to be fine-tuned according to different scenarios.

Table 4 Performance comparisons of different methods

Dataset	Douban-Book		MovieLens-1M		Yelp2018	
Method	**recall**	**ndcg**	**recall**	**ndcg**	**recall**	**ndcg**
NGCF	0.1380(-0.86%)	0.1165(-1.93%)	0.0812(-0.31%)	0.1587(-0.99%)	0.0612(-4.07%)	0.0502(-3.46%)
LightGCN	0.1392	0.1188	0.0838	0.1603	0.0638	0.0520
SGL-ND	0.1626(+16.81%)	0.1450(+22.05%)	0.0856(+2.15%)	0.1667(+3.99%)	0.0643(+0.78%)	0.0526(+1.15%)
SGL-ED	0.1732(+24.43%)	0.1549(+30.39%)	0.0864(+3.10%)	0.1656(+3.31%)	0.0675(+5.79%)	0.0555(+6.73%)
SGL-RW	0.1731(+24.35%)	0.1545(+30.05%)	0.0852(+1.67%)	0.1645(+2.62%)	0.0667(+4.54%)	0.0547(+5.19%)
SelfCF-HE	0.1742(+25.14%)	0.1569(+32.07%)	0.0889(+6.09%)	0.1684(+5.05%)	0.0661(+3.61%)	0.0542(+4.23%)
SelfCF-ED	0.1401(+0.65%)	0.1223(+2.95%)	0.0833(-0.60%)	0.1610(+0.44%)	0.0639(+0.16%)	0.0521(+0.19%)
SelfCF-EP	0.1365(-1.94%)	0.1152(-3.03%)	0.0729(-13.01%)	0.1492(-6.92%)	0.0640(+0.31%)	0.0531(+2.12%)
AFGRL	0.1654(+18.82%)	0.1254(+20.29%)	0.0869(+3.70%)	0.1632(+1.81%)	0.0702(+10.03%)	0.0553(+6.35 %)
SimGCL	0.1770(+27.16%)	0.1582(+33.16%)	0.0887(+5.85%)	0.1695(+5.74%)	0.0721(+13.01%)	0.0596(+14.62%)
SimSGR	**0.1904(+36.78%)**	**0.1624(+36.70%)**	**0.0905(+8.00%)**	**0.1701(+6.11%)**	**0.0742(+16.30%)**	**0.0602(+15.77%)**

Fig. 4 The performance of SimSGR with varied combinations of positive sets and aggregation methods

The performance of conversion layer

Given the positive pair Z and Z', we first convert them into the corresponding rating matrix. Meanwhile, the IC loss is designed to maintain the consistency of the corresponding terms in the rating matrix. To verify the effectiveness of the conversion layer, we modify the invariance term of IC loss as IC-variants (IC_V):

$$s(Z, Z') = \frac{1}{n} \sum_i \|z_i - z'_i\|_2^2. \tag{11}$$

We directly pull the positives in representation space, which is consistent with the traditional CL task. For a more comprehensive comparison, we introduce two other classic self-supervised loss functions: InfoNCE (IN) [5] and Barlow Twins (BT) [11]. Finally, we adopt the loss functions above (including IC) to optimize our model under the training settings of CL-only and joint learning separately. The Comparison is shown in Fig. 5:

1) Whether jointly optimizing the CL task and the main task of the recommendation or only optimizing the CL task, Our IC outperforms the others, which proves that the loss function in this paper is beneficial to improving the performance of the recommendation.
2) In the setting of only training the CL task, our IC outperforms others in a wide range, mainly due to the conversion layer that projects the positives of repre-

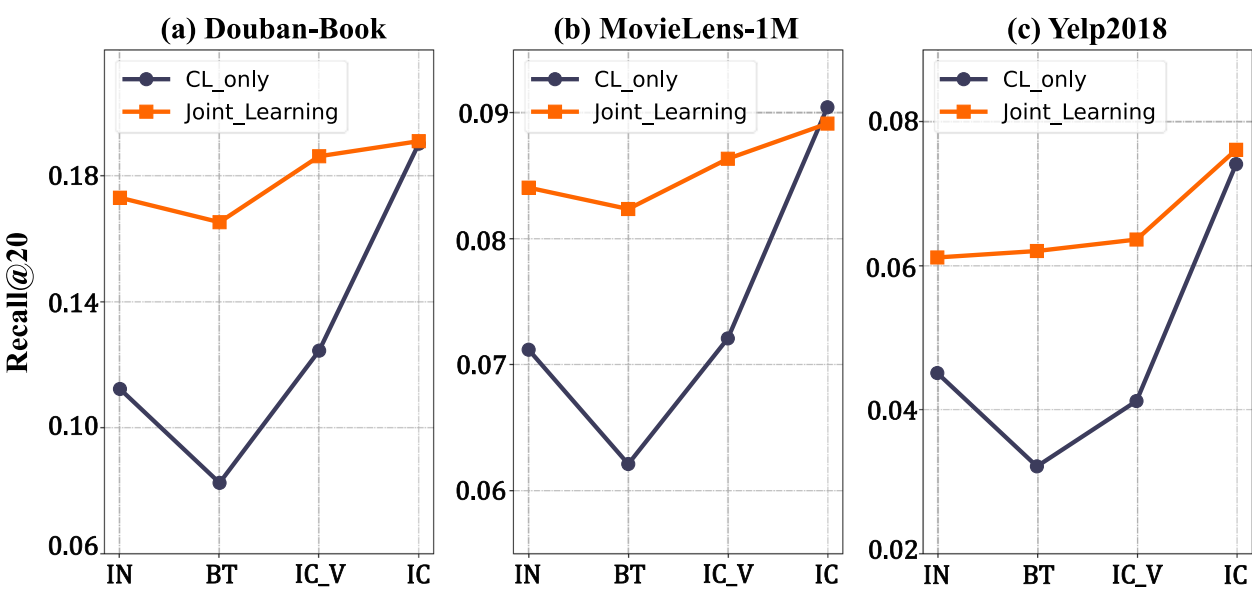

Fig. 5 The performance of SimSGR with different loss functions

sentation space into the rating space, unifying the CL task and the main task of recommendation system. At this time, the training no longer depends on the main task of the recommendation, which improves the performance and simplifies the framework.

3) Adopting the joint learning scheme has no obvious improvement for IC. We argue that this is mainly because IC has covered the main task of the recommendation system and the extra training term is redundant.

The performance in cold-start environments

SSL can effectively alleviate the data sparseness problem called cold-start in recommendation due to the ability to generate supervisory signals through the data itself. We conduct the following experiments to verify the effect of SimSGR on the cold-start problem.

In order to simulate the real cold-start scenario, for each user, inspired by [33], we filter the N items that have recently interacted with through the timestamp, where N is the threshold that determines the cold-start degree of the recommendation. Figure 6 shows the performance of various models on $N = 5$ (Cold-5), $N = 10$ (Cold-10), and the raw dataset (Raw, note that the number of interactions per user in the Raw may exceed 10, while the number of interactions per user in the cold-10 fix to 10).

Across the Raw, Cold-10, and Cold-5 settings, LightGCN achieved recall values of 0.1492, 0.1221, and 0.1021. SGL-ED and SimGCL achieved Recall values of 0.1732, 0.1646, and 0.1523, and 0.1770, 0.1721, and 0.1669, respectively. Meanwhile, the recall values of SimSGR are 0.1951, 0.1906, and 0.1867. As the degree of cold start deepens, LightGCN's performance decreased by 18.12% and 16.56%, while SGL-ED's performance declined by 5.00% and 7.51%, and SimGCL's performance decreased by 2.75% and 3.02%, respectively. SimSGR also exhibited a decline in performance of 2.44% and 2.05%, respectively.

In conclusion, the performance of the three methods SGL, SimGCL, and SimSGR introduced SSL strategy has sightly declined on three datasets, but the overall performance maintains stable compared to LightGCN. We argue that since the supervision signal of LightGCN merely comes from the historical interaction, it is more susceptible to data sparseness.

In all cold-start scenarios, SimSGR consistently outperforms others. We argue that the Mixing and Conversion mechanisms are the determining factors of SimSGR, which make it possible to generate vast credible positive pairs without relying on real labels.

The impact of the hyperparameter

This section explores the effects of hyperparameters K_{pos} in SimSGR, which controls the size of the final positive set before mixing. We vary K_{pos} in {4, 8, 16, 32}.

The results in Fig. 7 indicates that SimSGR achieves the best performance when $K_{pos} = 4$ in Douban-Book and Yelp2018. Meanwhile, the model performs best when $K_{pos} = 8$ in MovieLens-1 M. This difference may be attributed to the high density of the MovieLens-1 M

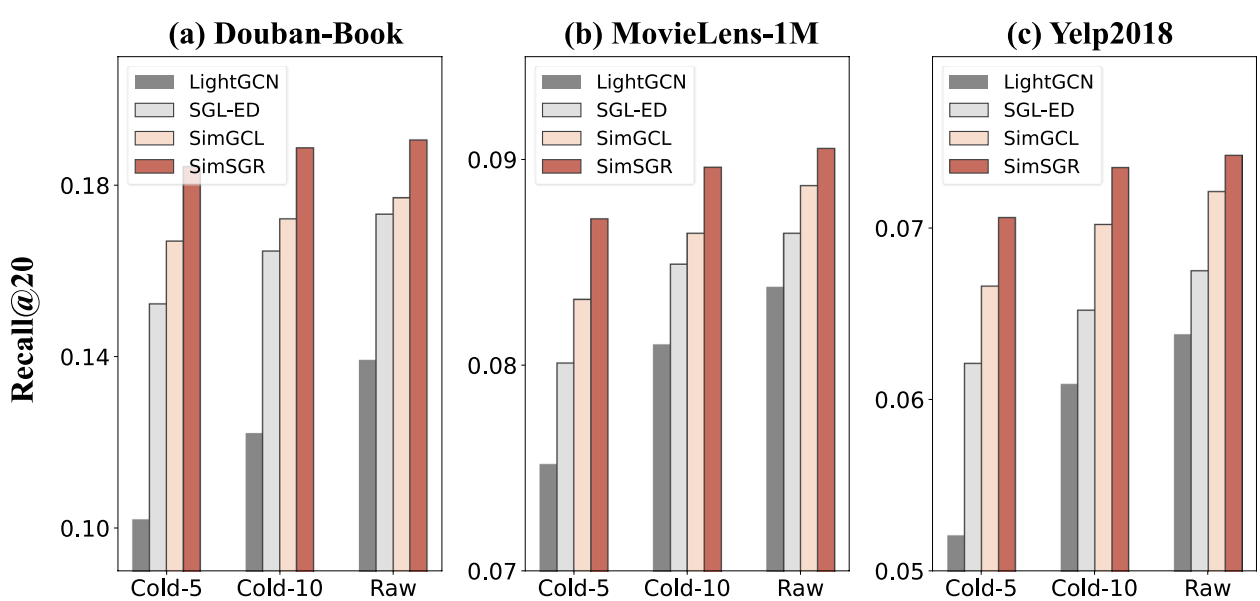

Fig. 6 The performance of SimSGR over the different cold-start degree

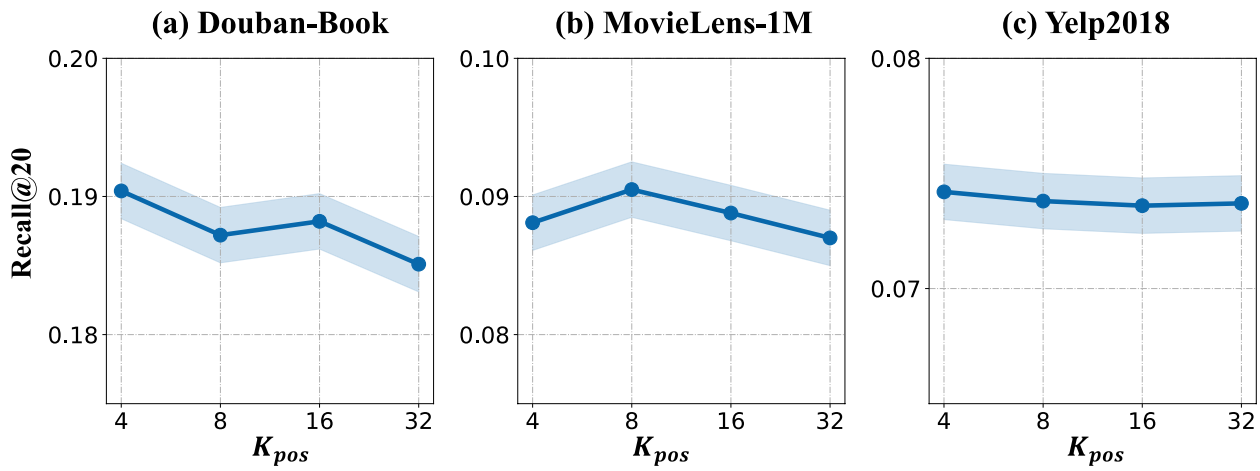

Fig. 7 The performance of SimSGR with varied values of K_{pos}

compared to the other two datasets, thus, more prior information can be obtained when the candidate positive samples increase. It is worth noting that, no matter on which dataset, the performance is relatively stable over varied K_{pos}, which verifies that our model is not sensitive to hyper-parameters and can be easily tuned.

Conclusion

In this paper, we adopted SSL into caching strategies and proposed a simple self-supervised graph-based recommendation framework for edge caching networks. Instead of randomly generating two augmented views from the user-item graph and directly regarding their corresponding representation as positives, SimSGR reconstructs positives through two layers: the Mixing layer and the Conversion layer. In the Mixing layer, we mixed the candidate nodes selected based on representation similarity and the topology of the graph to generate credible positive pairs. In the Conversion layer, we calculated the prediction score matrix, which converts positives into corresponding elements in the two rating matrices, leading to the unification of the SSL task and the main task of the recommendation system. These modifications simplify the model structure and make it more suitable for edge caching networks with resource limitations. Comprehensive experiments have been conducted on various real-world datasets, demonstrating that our SimSGR consistently outperforms existing state-of-art methods, especially in cold-start environments.

Authors' contributions

Sun Aijing provided the research direction and innovation points of this paper, Wang Guoqing was mainly responsible for algorithm implementation and code writing, Qi Han mainly provided hardware support and English polishing and all authors participated in the manuscript writing.

Funding

We would like to thank the anonymous reviewers for their valuable comments. The publication of the article is supported by the Natural Science Foundation of China under Grant 61901367.

Availability of data and materials

The datasets are available online. The URL is as follows:
Douban-Book: https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding;
MovieLens-1 M: https://github.com/RUCAIBox/RecSysDatasets/tree/master/dataset_info/MovieLens;
Yelp2018: https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018.

Declarations

Ethics approval and consent to participate

This declaration is not applicable.

Competing interests

The authors declare no competing interests.

Received: 17 February 2023 Accepted: 1 July 2023
Published online: 22 July 2023

References

1. Du J, Yu FR, Lu G, Wang J, Jiang J, Chu X (2020) MEC-assisted immersive VR video streaming over terahertz wireless networks: a deep reinforcement learning approach. IEEE Internet of Things J 7(10):9517–9529
2. Shuja J, Bilal K, Alasmary W, Sinky H, Alanazi E (2021) Applying machine learning techniques for caching in next-generation edge networks: a comprehensive survey. J Netw Comput Appl 181:103005
3. Mao S, Liu L, Zhang N, Dong M, Zhao J, Wu J, Leung VC (2022) Reconfigurable intelligent surface-assisted secure mobile edge computing networks. IEEE Trans Veh Technol 71:6647–60
4. Du J, Cheng W, Guangyue Lu, Cao H, Chu X, Zhang Z, Wang J (2021) Resource pricing and allocation in MEC enabled blockchain systems: an A3C deep reinforcement learning approach. IEEE Trans Netw Sci Eng 9(1):33–44
5. Wei X, Liu J, Wang Y, Tang C, Yongyang Hu (2021) Wireless edge caching based on content similarity in dynamic environments. J Syst Architect 115:102000

6. Chang Z, Lei L, Zhou Z, Mao S, Ristaniemi T (2018) Learn to cache: machine learning for network edge caching in the big data era. IEEE Wirel Commun 25(3):28–35
7. Feng J, Liu L, Pei Q, Li K (2021) Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks. IEEE Trans Parallel Distrib Syst 33(11):2687–2700
8. Hao B, Zhang J, Yin H et al (2021) Pre-training graph neural networks for cold-start users and items representation. Proceedings of the 14th ACM International Conference on Web Search and Data Mining. pp 265–273
9. Yu J, Yin H, Xia X, et al (2022) Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. arXiv preprint arXiv:2112.08679
10. Wu J, Wang X, Feng F et al (2021) Self-supervised graph learning for recommendation. Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. pp 726–735
11. Lee N, Lee J, Park C (2021) Augmentation-free self-supervised learning on graphs. arXiv preprint arXiv:2112.02472
12. Feng J, Zhang W, Pei Q, Jinsong Wu, Lin X (2022) Heterogeneous computation and resource allocation for wireless powered federated edge learning systems. IEEE Trans Commun 70(5):3220–3233
13. Zbontar J, Jing L, Misra I, et al (2021) Barlow twins: Self-supervised learning via redundancy reduction[C]//International Conference on Machine Learning. PMLR: 12310–12320
14. Velickovic P, Fedus W, Hamilton WL et al (2019) Deep Graph Infomax. ICLR (Poster) 2(3):4
15. Hassani K, Khasahmadi AH (2020) Contrastive multi-view representation learning on graphs[C]//International Conference on Machine Learning. PMLR: 4116–4126
16. Qiu J, Chen Q, Dong Y et al (2020) Gcc: Graph contrastive coding for graph neural network pre-training. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp 1150–1160
17. Zhu Y, Xu Y, Yu F, et al (2020) Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131
18. You Y, Chen T, Sui Y et al (2020) Graph contrastive learning with augmentations. Adv Neural Inf Process Syst 33:5812–5823
19. Zhu Y, Xu Y, Yu F et al (2021) Graph contrastive learning with adaptive augmentation. Proceedings of the Web Conference 2021. pp 2069–2080
20. Chen T, Kornblith S, Norouzi M, et al (2020) A simple framework for contrastive learning of visual representations[C]//International conference on machine learning. PMLR: 1597–1607
21. He K, Fan H, Wu Y et al (2020) Momentum contrast for unsupervised visual representation learning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp 9729–9738
22. Grill JB, Strub F, Altché F et al (2020) Bootstrap your own latent-a new approach to self-supervised learning. Adv Neural Inf Process Syst 33:21271–21284
23. Bardes A, Ponce J, LeCun Y (2021) Vicreg: Variance-invariance-covariance regularization for self-supervised learning. arXiv preprint arXiv:2105.04906
24. Zhou K, Wang H, Zhao WX et al (2020) S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp 1893–1902
25. Yu J, Yin H, Gao M et al (2021) Socially-aware self-supervised tri-training for recommendation. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp 2084–2092
26. Xie X, Sun F, Liu Z, et al (2020) Contrastive learning for sequential recommendation. arXiv preprint arXiv:2010.14395
27. Liu Z, Chen Y, Li J, et al (2021) Contrastive self-supervised sequential recommendation with robust augmentation. arXiv preprint arXiv:2108.06479
28. He X, Deng K, Wang X et al (2020) Lightgcn: Simplifying and powering graph convolution network for recommendation. Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. pp 639–648
29. Mao K, Zhu J, Wang J et al (2021) SimpleX: A Simple and Strong Baseline for Collaborative Filtering. Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp 1243–1252
30. Zhao WX, Mu S, Hou Y et al (2021) Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp 4653–4664
31. Wang X, He X, Wang M et al (2019) Neural graph collaborative filtering. Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. pp 165–174
32. Zhou X, Sun A, Liu Y, et al (2021) SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. arXiv preprint arXiv:2107.03019, vvvv
33. Zhang Y, Shi Z, Zuo W, et al (2020) Joint Personalized Markov Chains with social network embedding for cold-start recommendation[J]. Neurocomputing 386:208–220

Publisher's Note