arXiv:2010.10783v1 [cs.IR] 21 Oct 2020

Self-supervised Graph Learning for Recommendation

Jiancan Wu
wjc1994@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Xiang Wang
xiangwang@u.nus.edu
National University of Singapore
Singapore

Fuli Feng
fulifeng93@gmail.com
National University of Singapore
Singapore

Xiangnan He
xiangnanhe@gmail.com
University of Science and Technology of China
Hefei, Anhui, China

Liang Chen
chenliang6@mail.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Jianxun Lian
Jianxun.Lian@microsoft.com
Microsoft Research Asia
China

Xing Xie
xing.xie@microsoft.com
Microsoft Research Asia
China

ABSTRACT

Representation learning on user-item graph for recommendation has evolved from using single ID or interaction history to exploiting higher-order neighbors. This leads to the success of graph convolution networks (GCNs) for recommendation such as PinSage [48] and LightGCN [17]. Despite effectiveness, we argue that they suffer from two limitations: (1) high-degree nodes exert larger impact on the representation learning, deteriorating the recommendations of low-degree (long-tail) items; and (2) representations are vulnerable to noisy interactions, as the neighborhood aggregation scheme further enlarges the impact of observed edges.

In this work, we explore self-supervised learning on user-item graph, so as to improve the accuracy and robustness of GCNs for recommendation. The idea is to supplement the classical supervised task of recommendation with an auxiliary self-supervised task, which reinforces node representation learning via self-discrimination. Specifically, we generate multiple views of a node, maximizing the agreement between different views of the same node compared to that of other nodes. We devise four operators to generate the views — embedding masking, embedding dropout, node dropout, and edge dropout — that augment node representation from two perspectives of ID embedding and graph structure. We term this new learning paradigm as *Self-supervised Graph Learning* (SGL), implementing it on the state-of-the-art model LightGCN. Empirical studies on three benchmark datasets demonstrate the effectiveness of SGL, which improves the recommendation accuracy, especially on long-tail items, and the robustness against interaction noises.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Collaborative filtering, Graph Neural Network, Self-supervised Learning, Long-tail

1 INTRODUCTION

Learning high-quality user and item representations from user-item interaction data is the theme of collaborative recommendation. Earlier work like matrix factorization (MF) [32] projects single ID of each user (or item) into an embedding vector. Some follow-on studies [18, 23] enrich the single ID with interaction history for learning better representations. More recently, representation learning has evolved from using single ID and interaction history to exploiting higher-order connectivity in user-item graph. The technique is inspired from the graph convolution networks (GCNs), which provide an effective end-to-end way to integrate multi-hop neighbors into node representation learning and achieve state-of-the-art performance for recommendation [17, 36, 44, 48].

Despite effectiveness, current GCN-based recommendation models suffer from some limitations:

- **Sparse Supervision Signal.** Most models approach the recommendation task under a supervised learning paradigm [17, 19, 32], where the supervision signal comes from the observed user-item interactions. However, the observed interactions are extremely sparse [3, 16] compared to the whole interaction space, making it insufficient to learn quality representations.
- **Skewed Data Distribution.** Observed interactions usually follow a power-law distribution [7, 30], where the long tail consists of low-degree items that lack supervision signal. In contrast, high-degree items appear more frequently in neighborhood aggregation and supervised loss, thus exert larger impact on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

the representation learning. Hence, the GCNs are easily biased towards high-degree items [34], sacrificing the performance of low-degree (long-tail) items.

- **Noises in Interactions.** Most feedback that a user provides is implicit (*e.g.,* clicks, views), instead of explicit (*e.g.,* ratings, likes/dislikes). As such, observed interactions usually contain noises, *e.g.,* a user is misled to click an item and finds it uninteresting after consuming it. [3, 22]. The neighborhood aggregation scheme in GCNs enlarges the impact of the interactions in representation learning, making the learning more vulnerable to interaction noises.

In this work, we focus on exploring self-supervised learning (SSL) in recommendation, to solve the foregoing limitations. Though being prevalent in computer vision (CV) [9, 25, 38] and natural language processing (NLP) [8, 24], SSL is relatively less explored in recommendation. The idea is to set an auxiliary task that distills supervision signal from the input data itself, especially through exploiting the unlabeled data space. For example, BERT [8] randomly masks some tokens in a sentence, setting the prediction of the masked tokens as the auxiliary task that can capture the dependencies among tokens; RotNet [9] randomly rotates labeled images, training the model on the rotated images to get improved representations for the mask task of object recognition or image classification. Clearly, compared with supervised learning, SSL allows us to exploit the unlabeled data space via making changes on the input labeled data, achieving remarkable improvements in downstream tasks [5].

Here we wish to bring the SSL's superiority into recommendation representation learning, which differs from CV/NLP tasks since the data are discrete and inter-connected. To address the argued limitations of GCN-based recommendation models, we construct the auxiliary task as discriminating the representation of a node itself. Specifically, it consists of two key components: (1) **data augmentation**, which generates multiple views for each node, and (2) **contrastive learning**, which maximizes the agreement between different views of the same node, compared to that of other nodes. By interpreting GCN as propagating node embeddings on the graph, we identify two variables as the model input: embedding matrix, which encodes the intrinsic characteristics of a node, and graph adjacency matrix, which presents the graph structure of user behaviors. From this view, we construct the unlabeled data space by changing the input, developing four operators to this end: embedding masking, embedding dropout, node dropout, and edge dropout. Each of the operators augments the data with a different purpose and rationality. Thereafter, the contrastive loss is employed to distinguish each node based on these augmented representations from the other nodes. As a result, SGL enables the discovery of sound representations by exploring the internal relationship among nodes.

Conceptually, our SGL supplements existing GCN-based recommendation models in: (1) node self-discrimination offers auxiliary supervision signal, which is complementary to the classical supervisions from observed interactions for representation learning; (2) the augmentation operators, especially edge dropout, helps to mitigate the degree biases by changing the graph structure and reduce the influence of high-degree nodes; and (3) node dropout and edge dropout create multiple views for nodes *w.r.t.* different local structures and neighborhoods, enhancing the model robustness against interaction noises.

It is worthwhile mentioning that our SGL is model-agnostic and can be applied to any model that consists of user embedding and item embedding. Here we implement it on a state-of-the-art GCN-based model, LightGCN [17]. Experimental studies on three benchmark datasets demonstrate the effectiveness of SGL, which significantly improves the recommendation accuracy, especially on long-tail items, and enhance the robustness against interaction noises. We summarize the contributions of this work as follows:

- To the best of our knowledge, this is the first work that develops self-supervised learning for graph-based recommendation.
- We devise a new learning paradigm, SGL, which takes node self-discrimination as the self-supervised task to offer auxiliary supervision signal for node representation learning.
- We conduct extensive experiments on three benchmark datasets to demonstrate the superiority of SGL.

2 PRELIMINARIES

We first summarize the common paradigm of GCN-based collaborative filtering models. Let $\mathcal{U}$ and $\mathcal{I}$ be the set of users and items respectively. Let $\mathcal{O}^+ = \{y_{ui} | u \in \mathcal{U}, i \in \mathcal{I}\}$ be the observed interactions between users and items, where y_{ui} indicates that user u has adopted item i before. Most existing models [17, 36, 44] construct a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ involves all users and items, and the edge set $\mathcal{E} = \mathcal{O}^+$ represents observed interactions.

Abstract Paradigm of GCN. At the core is to apply the neighborhood aggregation scheme on $\mathcal{G}$, recursively integrate the vectorized information from neighboring nodes, and update the representations of ego nodes. When only ID information is available as the pre-existing feature of each user (or item), a widely-used solution is to parameterize each ID with an embedding vector. As such, the overall scheme is formulated as follows:

$$\mathbf{Z} = H(\mathbf{E}, \mathcal{G}) \tag{1}$$

where $H(\cdot)$ is the GCN function to encode connectivity information into representation learning; $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the final representations of all nodes, while $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d'}$ is the ID embedding matrix of all nodes; d and d' denote the size of final representations and ID embeddings, respectively. Clearly, there are two key factors affecting the representation learning: (1) ID embeddings, which reflect the intrinsic characteristics of users and items; and (2) graph structure, which offers the structural information — *i.e.,* the underlying connectivity (paths) among users and items.

Neighborhood Aggregation. The aggregation scheme consists of two crucial components:
(1) Representation aggregation layers. After l layers, a node's representation is able to capture the structural information within its l-hop neighbors. The l-th layer is formulated as:

$$\begin{aligned} \mathbf{a}_u^{(l)} &= f_{\text{aggregate}}\Big(\{\mathbf{z}_i^{(l-1)} | i \in \mathcal{N}_u\}\Big), \\ \mathbf{z}_u^{(l)} &= f_{\text{combine}}(\mathbf{z}_u^{(l-1)}, \mathbf{a}_u^{(l)}), \end{aligned} \tag{2}$$

where $\mathbf{a}_u^{(l)}$ denotes the aggregation of vectorized information from node u's neighborhood $\mathcal{N}_u$; and $\mathbf{z}_u^{(l)}$ is the representation of node u after l aggregation layers, which integrates $\mathbf{a}_u^{(l)}$ with its previous representation $\mathbf{z}_u^{(l-1)}$. We initiate $\mathbf{z}_u^{(0)}$ with ID embedding $\mathbf{e}_u$. There are a number of designs for $f_{\text{aggregate}}(\cdot)$ and $f_{\text{combine}}(\cdot)$ [10, 13, 39, 46].

(2) Readout layer. Having obtained the representations at different layers, the readout function generates the final representations:

$$\mathbf{z}_u = f_{\text{readout}}\Big(\{\mathbf{z}_u^{(l)} | l = [0, \cdots, L]\}\Big), \tag{3}$$

which can be simply set as the last-layer representation [36, 48], concatenation [44], or summation [17] over the representations of all layers.

Supervised Learning Loss. Thereafter, a prediction layer is built upon the final representations of user u and item i, to predict how likely u would adopt i. A classical solution is the inner product, which supports fast top-k retrieval:

$$\hat{y}_{ui} = \mathbf{z}_u^\top \mathbf{z}_i. \tag{4}$$

To optimize model parameters, existing works usually frame the recommendation task as one of supervised learning. Hence, the observed interactions are the source of supervision signal, encouraging the predicted value $\hat{y}_{ui}$ to be close to the ground truth value y_{ui}. Whereas, the missing data serve as negative. Besides the point-wise classification loss [19], another intensively used loss in recommendation is the pairwise Bayesian Personalized Ranking (BPR) loss [32], which enforces the prediction of an observed interaction to be scored higher than its unobserved counterparts:

$$\mathcal{L}_{main} = \sum_{(u,i,j)\in O} -\log \sigma(\hat{y}_{ui} - \hat{y}_{uj}), \tag{5}$$

where $O = \{(u, i, j) | (u, i) \in O^+, (u, j) \in O^-\}$ is the training data, and $O^- = \mathcal{U} \times \mathcal{I} \setminus O^+$ is the unobserved interactions.

3 METHODOLOGY

We present the proposed Self-supervised Graph Learning (SGL) paradigm, which supercharges the supervised learning task (*cf.* Equation (5)) with self-supervised learning. Figure 1 illustrates the working flow of SGL. Specifically, the self-supervised task, also termed as pretext task or auxiliary task, is to construct supervision signal from the correlation within the input data. Such correlation information can supplement the main supervised task effectively.

Next, we introduce how to perform data augmentation that generates multiple representation views for a node, followed by the contrastive learning based on the generated representations to build the pretext task. SSL is combined with classical GCN in a multi-task learning manner. Lastly, we analyze the complexity of SSL.

3.1 Data Augmentation

Directly grafting the data augmentation adopted in CV and NLP tasks [5, 8, 15, 45] is infeasible for graph-based recommendation, due to specific characteristics: (1) The features of users and items are discrete, like one-hot ID and other categorical variables. Hence, the augmentation operators on images, such as random crop, rotation, or blur, are not suitable. (2) More importantly, unlike CV and NLP

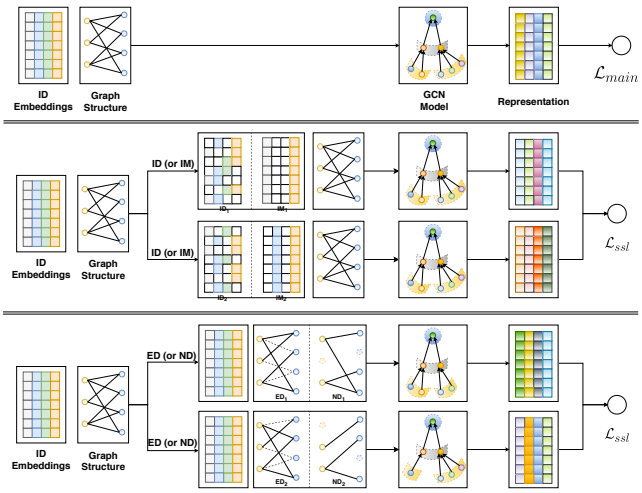


Figure 1: The overall system framework of SGL. (1) The first layer illustrates the working flow of the main supervised learning task. (2) The second and third layers show the working flows of SSL task with augmentation on ID embeddings and graph structure, respectively.

tasks that treat each data instance as isolated, users and items in the interaction graph are inherently connected and dependent of each others. Thus, we need new augmentation operators tailored for graph-based recommendation. According to our analysis in Section 2, we identify two inputs for GCN learning: ID embeddings and graph structure. Hence, we correspondingly design two types of augmentation, considering different aspects of the data space.

3.1.1 ***Augmentation on ID embeddings.*** Before aggregating the information from neighborhood, the ID embedding can describe the intrinsic characteristics of each node. Hence, we propose two augmentation operators, embedding masking and embedding dropout, to generate the augmented views of ID embeddings. The operators can be formally summarized as:

$$\mathbf{Z}' = H(t'(\mathbf{E}), \mathcal{G}), \quad \mathbf{Z}'' = H(t''(\mathbf{E}), \mathcal{G}), \quad t', t'' \sim \mathcal{T}, \tag{6}$$

where two stochastic data augmentation modules t' and t'' are applied on ID embedding matrix $\mathbf{E}$, and result in two correlated views of node representations $\mathbf{Z}'$ and $\mathbf{Z}''$. For each node u, we consider the augmented views $\mathbf{z}'_u$ and $\mathbf{z}''_u$ as one positive pair, otherwise the negative pair. In particular, t' and t'' can be formulated as:

$$t'(\mathbf{E}) = \mathbf{M}' \odot \mathbf{E}, \quad t''(\mathbf{E}) = \mathbf{M}'' \odot \mathbf{E}, \tag{7}$$

where $\mathbf{M}', \mathbf{M}'' \in \{0, 1\}^{|\mathcal{V}| \times d}$ are the masking matrix which are generated by embedding masking or embedding dropout strategy:

- **ID Embedding Masking (IM).** With the mask ratio ρ, a subset of dimensions are masked for all ID embeddings, so as to use partial dimensions to learn representations. By applying two independent masking processes t' and t'', this approach is able to focus on different dimensions and learn the internal relationship between two subsets of embeddings.
- **ID Embedding Dropout (ID).** With the dropout ratio ρ, a subset of elements in an embedding are set as 0, such that the

remaining elements are used to discriminate each node against others. Through two independent dropout processes t' and t'', this approach encourages partial information to recover the instance-wise embeddings, and learn not to heavily rely on certain information, so as to improve model robustness [41].

Note that the stochastic selections are easy to adapt to content-based recommendation models. Next, we consider the characteristics of graph structure.

3.1.2 ***Augmentation on Graph Structure.*** The bipartite graph is built upon observed user-item interactions, thus containing the collaborative filtering signal. Specifically, the first-hop neighborhood directly profiles ego user and item nodes — *i.e.,* historical items of a user (or interacted users of an item) can be viewed as the pre-existing features of user (or item). The second-hop neighboring nodes of a user (or an item) exhibit similar users *w.r.t.* behaviors (or similar items *w.r.t.* audiences). Furthermore, the higher-order paths from a user to an item reflect potential interests of the user on the item. Undoubtedly, mining the inherent patterns in graph structure is helpful to representation learning. We hence devise two operators, node dropout and edge dropout, to create different views of nodes. The operators can be formulated as follows:

$$\mathbf{Z}' = H(\mathbf{E}, s'(\mathcal{G})), \quad \mathbf{Z}'' = H(\mathbf{E}, s''(\mathcal{G})), \quad s', s'' \sim \mathcal{S}, \tag{8}$$

where two stochastic selections s' and s'' are independently applied on graph $\mathcal{G}$, and establish two correlated views of nodes $\mathbf{Z}'$ and $\mathbf{Z}''$. We elaborate the augmentation operators as follows:

- **Node Dropout (ND).** With the probability ρ, each node is discarded from the graph, together with its connected edges. In particular, s' and s'' can be modeled as:

$$s'(\mathcal{G}) = (\mathbf{M}' \odot \mathcal{V}, \mathcal{E}), \quad s''(\mathcal{G}) = (\mathbf{M}'' \odot \mathcal{V}, \mathcal{E}), \tag{9}$$

  where $\mathbf{M}', \mathbf{M}'' \in \{0,1\}^{|\mathcal{V}|}$ are two masking vectors which are applied on the node set $\mathcal{V}$ to generate two subgraphs. As such, this augmentation is expected to identify the influential nodes from differently augmented views, and make the representation learning less sensitive to structure changes.
- **Edge Dropout (ED).** The augmentation drops out the edges in graph with a dropout ratio ρ. Two independent processes are represented as:

$$s'(\mathcal{G}) = (\mathcal{V}, \mathbf{M}' \odot \mathcal{E}), \quad s''(\mathcal{G}) = (\mathcal{V}, \mathbf{M}'' \odot \mathcal{E}), \tag{10}$$

  where $\mathbf{M}', \mathbf{M}'' \in \{0,1\}^{|\mathcal{E}|}$ are two masking vectors on the edge set $\mathcal{E}$. Only partial connections within the neighborhood contribute to the node representations. As such, coupling these two subgraphs together aims to capture the useful patterns of the local structures of a node, and further endows the representations more robustness against the presence of single interactions, especially the noisy interactions.

We apply these augmentations on ID embeddings or graph structure per epoch for simplicity — that is, we generate two different views of each node at the beginning of a new training epoch. Note that the dropout and masking ratios remain the same for two independent processes (*i.e.,* t' and t'', or s' and s''). We leaving the different ratios in future work. It is also worthwhile mentioning that only dropout and masking operations are involved, and no any model parameters are added.

Algorithm 1: Learning algorithm for SGL-ED

```
Input: Adjacency matrix of user-item graph, λ1, λ2, τ, ρ
while not converge do
    foreach epoch do
        Perform Eq. (8) for data augmentation
        foreach batch do
            Evaluate L_main according to Eq. (5)
            Evaluate L_self according to Eq. (11)
            Evaluate L according to Eq. (12)
            Update the parameters by gradient descent
        end
    end
end
```

3.2 Contrastive Learning

Having established the augmented views of nodes, we treat the views of the same node as the positive pairs (*i.e.,* $\{(\mathbf{z}'_u, \mathbf{z}''_u) | u \in \mathcal{U}\}$), and the views of any different nodes as the negative pairs (*i.e.,* $\{(\mathbf{z}'_u, \mathbf{z}''_v) | u, v \in \mathcal{U}, u \neq v\}$). The auxiliary supervision of positive pairs encourages the consistency between different views of the same node for prediction, while the supervision of negative pairs enforces the divergence among different nodes. Formally, we follow SimCLR [5] and adopt the contrastive loss, InfoNCE [12], to maximize the agreement of positive pairs and minimize that of negative pairs:

$$\mathcal{L}_{ssl}^{user} = \sum_{u \in \mathcal{U}} -\log \frac{\exp(s(\mathbf{z}'_u, \mathbf{z}''_u)/\tau)}{\sum_{v \in \mathcal{U}} \exp(s(\mathbf{z}'_u, \mathbf{z}''_v)/\tau)}, \tag{11}$$

where $s(\cdot)$ measures the similarity between two vectors, which is set as cosine similarity function; τ is the temperature hyperparameter. Analogously, we obtain the contrastive loss of the item side $\mathcal{L}_{ssl}^{item}$. Combining these two losses, we get the objective function of self-supervised task as $\mathcal{L}_{ssl} = \mathcal{L}_{ssl}^{user} + \mathcal{L}_{ssl}^{item}$.

3.3 Multi-task Training

To improve recommendation with the SSL task, we leverage a multi-task training strategy to jointly optimize the classic recommendation task (*cf.* Equation (5)) and the self-supervised learning task (*cf.* Equation (11))

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_{ssl} + \lambda_2 \|\Theta\|_2^2, \tag{12}$$

where Θ is the set of model parameters in L_{main} since L_{ssl} introduces no additional parameters; λ_1 and λ_2 are hyperparameters to control the strengths of SSL and L_2 regularization, respectively. The overall learning algorithm of SGL equipped with ED (termed SGL-ED) is summarized in Algorithm 1. Other implementations (*i.e.,* , SGL-ID, SGL-IM, SGL-ND) follow similar workflow. We also consider the alternative optimization — pre-training on $\mathcal{L}_{ssl}$ and fine-tuning on $\mathcal{L}_{main}$. See more details in Section 4.4.5.

3.4 Model Complexity Analyses

In this subsection, we analyze the complexity of SGL with ED as the strategy and LightGCN as the recommendation model; other choices can be analyzed similarly. Since SGL introduces no

Table 1: The comparison of analytical time complexity between LightGCN and SGL-ED.

Component	LightGCN	SGL-ED
Adjacency Matrix	$O(2\,\lvert E\rvert)$	$O(4\hat{\rho}\,\lvert E\rvert\, s + 2\,\lvert E\rvert)$
Graph Convolution	$O(2\,\lvert E\rvert\, Lds\frac{\lvert E\rvert}{B})$	$O(2(1+2\hat{\rho})\,\lvert E\rvert\, Lds\frac{\lvert E\rvert}{B})$
BPR Loss	$O(2\,\lvert E\rvert\, ds)$	$O(2\,\lvert E\rvert\, ds)$
Self-supervised Loss	-	$O(\lvert E\rvert\, d(2+\lvert V\rvert)s)$
		$O(\lvert E\rvert\, d(2+2B)s)$

trainable parameters, the space complexity remains the same as LightGCN [17]. The time complexity of model inference is also the same, since there is no change on the model structure. In the following part, we will analyze the time complexity of SGL training.

Suppose the number of nodes and edges in the user-item interaction graph are $|V|$ and $|E|$ respectively. Let s denote the number of epochs, B denote the size of each training batch, d denote the embedding size, L denote the number of GCN layers, $\hat{\rho} = 1 - \rho$ denote the keep probability of SGL-ED. The complexity mainly comes from two parts:

- Normalization of adjacency matrix. Since we generate two independent sub-graphs per epoch, given the fact that the number of non-zero elements in the adjacency matrices of full training graph and two sub-graph are $2\,|E|$, $2\hat{\rho}\,|E|$ and $2\hat{\rho}\,|E|$ respectively, its total complexity is $O(4\hat{\rho}\,|E|\, s + 2\,|E|)$.
- Evaluating self-supervised loss. We only consider the inner product in our analyses. As defined in Equation (11), we treat all other user nodes as negative samples when calculating InfoNCE loss of user side. Within a batch, the complexity of numerator and denominator are $O(Bd)$ and $O(BMd)$, respectively, where M is the number of users. And hence the total complexity of both user and item side per epoch is $O(|E|\, d(2+|V|))$. Therefore, the time complexity of the whole training phase is $O(|E|\, d(2+|V|)s)$. An alternative to reduce the time complexity is treating only the users (or the items) within the batch as negative samples [5, 47], resulting in total time complexity of $O(|E|\, d(2+2B)s)$.

We summarize the time complexity in training between LightGCN and SGL-ED in Table 1. The analytical complexity of LightGCN and SGL-ED is actually in the same magnitude, since the increase of LightGCN only scales the complexity of LightGCN. In practice, taking the Yelp2018 data as an example, the time complexity of SGL-ED (alternative) with $\hat{\rho}$ of 0.8 is about 3.7x larger than LightGCN, which is totally acceptable considering the speedup of convergence speed we will show in Section 4.3.2. We implement our SGL in TensorFlow and will release our codes upon acceptance. The testing platform is Nvidia Titan RTX graphics card equipped with Inter i7-9700K CPU (32GB Memory). The time cost of each epoch on Yelp2018 is 15.2s and 60.6s for LightGCN and SGL-ED (alternative) respectively, which is consistent with the complexity analyses.

4 EXPERIMENTS

To justify the superiority of SGL and reveal the reasons of its effectiveness, we conduct extensive experiments to answer the following research questions:

Table 2: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062
Alibaba-iFashion	300,000	81,614	1,607,813	0.00007

- **RQ1:** As compared with the state-of-the-art CF models, how does SGL perform *w.r.t.* top-k recommendation?
- **RQ2:** What are the benefits of performing self-supervised learning in collaborative filtering?
- **RQ3:** How do different settings influence the effectiveness of the proposed SGL?

4.1 Experimental Settings

We conduct experiments on three widely used benchmark datasets: Yelp2018[17, 44], Amazon-Book[17, 44], and Alibaba-iFashion [6][1]. Following [17, 44], we use the same 10-core setting for Yelp2018 and Amazon-Book. Alibaba-iFashion is more sparse, where we randomly sample 300k users and use all their interactions over the fashion outfits. The statistics of all three datasets are summarized in Table 2. We follow the same strategy described in [44] to split the interactions into training, validation, and testing with a ratio of 7:1:2. For users in the testing set, we follow the all-ranking protocol [44] to evaluate the top-K recommendation performance and report the average Recall@K and NDCG@K where we set $K = 20$.

4.1.1 ***Compared Methods.*** We compare the proposed SGL with the following CF models:

- NGCF [44]. This is a graph-based CF method largely follows the standard GCN [10], including the use of nonlinear activation and feature transformation. Besides, it additionally encodes the second-order feature interaction into the message during message passing. We tune the regularization coefficient λ_2 and the number of GCN layers within the suggested ranges.
- LightGCN [17]. This is the state-of-the-art graph-based CF method which devises a light graph convolution to ease the training difficulty and pursue better generation ability. Similarly, we tune the λ_2 and the number of GCN layers.
- Mult-VAE [27]. This is an item-based CF method based on the variational auto-encoder (VAE). It uses multinomial likelihood to model user-item implicit feedback data and learns latent-variable model with variational inference. The model is optimized with an additional reconstruction objective, which can be seen as a special case of SSL. We follow the suggested model setting and tune the dropout ratio and β.

We discard potential baselines like MF [32], NeuMF [19], GC-MC [36], and PinSage [48] since the previous work [17, 27, 44] has validated the superiority over the compared ones. Upon LightGCN, we implement four variants of the proposed SGL, named SGL-ID, SGL-IM, SGL-ND and SGL-ED, which are equipped with ID Embedding Dropout, ID Embedding Masking, Node Dropout and Edge Dropout, respectively.

4.1.2 ***Hyper-parameter Settings.*** For fair comparison, all models are trained from scratch which are initialized with the Xavier method [11]. The models are optimized by the Adam optimizer with

[1] https://github.com/wenyuer/POG

Table 3: Performance comparison with LightGCN at different layers. The performance of LightGCN on Yelp2018 and Amazon-Book are copied from its original paper. The percentage in brackets denote the relative performance improvement over LightGCN. The bold indicates the best result, while the second-best performance is underlined.

Dataset		Yelp2018		Amazon-Book		Alibaba-iFashion	
#Layer	Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
1 Layer	LightGCN	0.0631	0.0515	0.0384	0.0298	0.0990	0.0454
	SGL-ID	0.0634(+0.5%)	0.0518(+0.6%)	0.0417(+8.6%)	0.0322(+8.1%)	**0.1141(+15.3%)**	**0.0540(+18.9%)**
	SGL-IM	0.0631(+0%)	0.0513(-0.4%)	0.0429(11.7%)	0.0331(+11.1%)	0.1116(+12.7%)	0.0530(+16.7%)
	SGL-ND	**0.0643(+1.9%)**	**0.0529(+2.7%)**	0.0432(+12.5%)	0.0334(+12.1%)	0.1133(+14.4%)	0.0539(+18.7%)
	SGL-ED	0.0637(+1.0%)	0.0526(+2.1%)	**0.0451(+17.4%)**	**0.0353(+18.5%)**	0.1132(+14.3%)	0.0539(+18.7%)
2 Layers	LightGCN	0.0622	0.0504	0.0411	0.0315	0.1066	0.0505
	SGL-ID	0.0659(+5.9%)	0.0539(+6.9%)	0.0436(+6.1%)	0.0342(+8.6%)	0.1089(+2.2%)	0.0517(+2.4%)
	SGL-IM	0.0657(+5.6%)	0.0538(+6.7%)	0.0434(+5.6%)	0.0338(+7.3%)	0.1085(+1.8%)	0.0517(+2.4%)
	SGL-ND	0.0658(+5.8%)	0.0538(+6.7%)	0.0427(+3.9%)	0.0335(+6.3%)	**0.1106(+3.8%)**	**0.0526(+4.2%)**
	SGL-ED	**0.0668(+7.4%)**	**0.0549(+8.9%)**	**0.0468(+13.9%)**	**0.0371(+17.8%)**	0.1091(+2.3%)	0.0520(+3.0%)
3 Layers	LightGCN	0.0639	0.0525	0.0410	0.0318	0.1078	0.0507
	SGL-ID	0.0649(+1.6%)	0.0533(+1.5%)	0.0450(+9.8%)	0.0353(+11.0%)	0.1119(+3.8%)	0.0529(+4.3%)
	SGL-IM	0.0652(+2.0%)	0.0536(+2.1%)	0.0449(+9.5%)	0.0353(+11.0%)	0.1121(+4.0%)	0.0537(+5.9%)
	SGL-ND	0.0644(+0.8%)	0.0528(0.6%)	0.0440(+7.3%)	0.0346(+8.8%)	0.1126(4.5%)	0.0536(+5.7%)
	SGL-ED	**0.0675(+5.6%)**	**0.0555(+5.7%)**	**0.0478(+16.6%)**	**0.0379(+19.2%)**	**0.1126(+4.5%)**	**0.0538(+6.1%)**

learning rate of 0.001 and mini-batch size of 2048. The early stopping strategy is the same as NGCF and LightGCN. The proposed SGL methods inherit the optimal values of the shared hyper-parameters. For the unique ones of SGL, we tune three λ_1, τ, and ρ within the ranges of $\{0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$, $\{0.1, 0.2, 0.5, 1.0\}$, and $\{0, 0.1, 0.2, \cdots, 0.5\}$, respectively.

4.2 Performance Comparison (RQ1)

4.2.1 ***Comparison with LightGCN.*** Table 3 shows the result comparison between SGL and LightGCN. We find that:

- In most cases, four SGL outperforms LightGCN by a large margin, indicating the superiority of supplementing the recommendation task with self-supervised learning.
- In SGL family, SGL-ED achieves best performance in 12 out of 18 cases and second place in 4 cases, verifying the effectiveness of adopting edge dropout as data augmentation. We attribute this to the power of SGL-ED in capturing the inherent patterns in graph structure.
- Considering two graph structure-based augmentations, SGL-ND is relatively unstable than SGL-ED. For example, on Yelp2018 and Amazon-Book, the results of SGL-ED increases with layers go deeper; whereas, SGL-ND exhibits different patterns: rise-decline on Yelp and decline-rise on Amazon-Book. Node Dropout can be viewed as a special case of Edge Dropout, which discards edges around a few nodes. Hence, dropping high-degree nodes will dramatically change the graph structure, thus exerts influence than the information aggregation in representation learning and makes the training unstable.
- The improvements on Amazon-Book and Alibaba-iFashion are more significant than that on Yelp2018. This might be caused by the characteristics of datasets. Specifically, in Amazon-Book and Alibaba-iFashion, supervision signal from user-item interactions is too sparse to guide the representation learning in LightGCN. Benefiting from the self-supervised task, SGL obtains auxiliary supervisions to assist the representation learning.

Table 4: Overall Performance Comparison.

Dataset	Yelp2018		Amazon-Book		Alibaba-iFashion	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
NGCF	0.0579	0.0477	0.0344	0.0263	0.1043	0.0486
LightGCN	0.0639	0.0525	0.0411	0.0315	0.1078	0.0507
Mult-VAE	0.0584	0.0450	0.0407	0.0315	0.1041	0.0497
SGL-ED	**0.0675**	**0.0555**	**0.0478**	**0.0379**	**0.1126**	**0.0538**

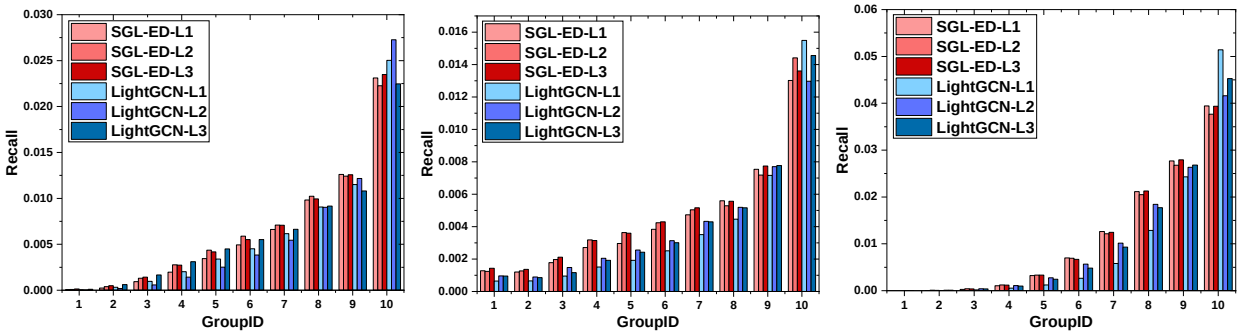
- Increasing the model depth from 1 layer to 3 layers is able to enhance the performance of SGL. This indicates that exploiting SSL could empower the generalization ability of GCN-based recommender models — that is, the contrastive learning among different nodes is of promise to solving the oversmoothing issue of node representations, further avoiding the overfitting problem of prediction.

4.2.2 ***Comparison with the State-of-the-Arts.*** In Table 4, we summarize the performance comparison with various baselines. As we can see, SGL-ED consistently outperforms other baselines across the board. This again demonstrates the rationality and effectiveness of incorporating the self-supervised learning. Among all the baselines, LightGCN achieves better performance than NGCF and Mult-VAE, which is consistent with the claim of LightGCN paper. The performance of Mult-VAE is on par with NGCF and Yelp on Alibaba-iFashion, while outperforms NGCF on Amazon-Book.

4.3 Benefits of SGL (RQ2)

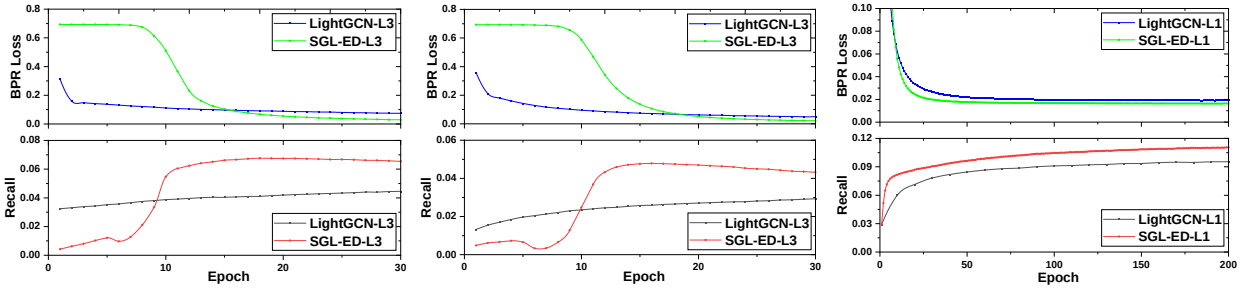
In this section, we study the benefits of SGL from three dimensions: (1) long-tail recommendation; (2) training efficiency; and (3) robustness to noises. Due to the limited space, we only report the results of SGL-ED, while having similar observations in the other methods.

4.3.1 ***Long-tail Recommendation.*** As Introduction mentions, GNN-based recommender models easily suffer from the long-tail problem. To verify whether SGL is of promise to solving the problem, we split items into ten groups based on the popularity,



(a) Yelp2018 (b) Amazon-Book (c) Alibaba-iFashion

Figure 2: Performance comparison over different item groups between SGL-ED and LightGCN. The suffix in the legend indicates the number of GCN layers.



(a) Yelp2018-L3 (b) Amazon-Book-L3 (c) Alibaba-iFashion-L1

Figure 3: Training Curves of SGL-ED and LightGCN on three datasets. The suffix in the legend denotes the layer numbers.

meanwhile keeping the total number of interactions of each group the same. The larger the GroupID is, the larger degrees the items have. We then decompose the Recall@20 metric of the whole dataset into contributions of single groups, as follows:

$$Recall = \frac{1}{m}\sum_{u=1}^{m}\frac{\sum_{g=1}^{10}\left|(l_{rec}^{u})^{(g)} \cap l_{test}^{u}\right|}{\left|l_{test}^{u}\right|} = \sum_{g=1}^{10} Recall^{(g)}$$

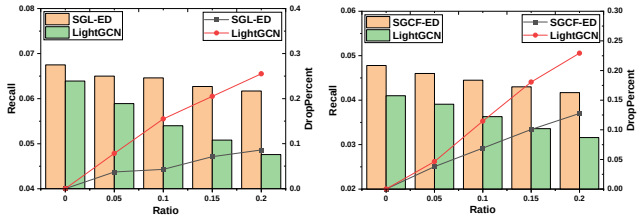
where m is the number of users, l_{rec}^{u} and l_{test}^{u} are the items in the top-N recommendation list and relevant items for user u, respectively. As such, $Recall^{(g)}$ measures the recommendation performance over the g-th group. We report the empirical results in 2 and find that:

- LightGCN is inclined to recommend high-degree items, while leaving long-tail items less exposed. Specifically, although only containing 0.83%, 0.83% and 0.22% of item spaces, the 10-th group contributes 39.72%, 39.92% and 51.92% of the total Recall scores in three datasets, respectively. This admits that, LightGCN hardly learns high-quality representations of long-tail items, due to the sparse interaction signal. Our SGL shows potentials in alleviating this issue: the contributions of the 10-group downgrade to 36.27%, 29.15% and 35.07% in three datasets, respectively.
- Jointly analyzing Table. 3 and Fig. 2, we find the performance improvements of SGL mainly come from accurately recommending the items with sparse interactions. This again verifies that the representation learning benefits greatly from auxiliary supervisions, so as to establish better representations of these items than LightGCN.

4.3.2 ***Training Efficiency.*** Self-supervised learning has proved its superiority in pre-training natural language model [8] and graph structure [21, 31]. Thus, we would like to study its influence on training efficiency. Fig. 3 shows the training curves of SGL-ED and LightGCN. As the number of epochs increases, the upper subfigures display the changes of training loss, while the bottoms record the performance changes in the testing set. We have the following observations:

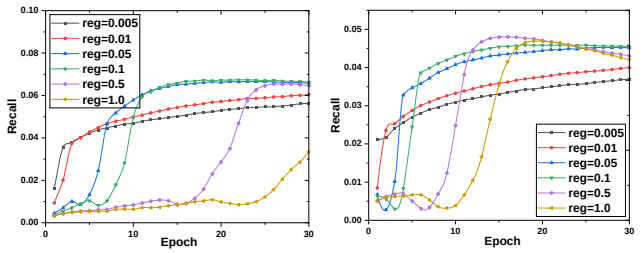
- Obviously, SGL is much faster to converge than LightGCN on Yelp2018 and Amazon-Book. In particular, SGL arrives at the best performance at the 18-th and 16-th epochs, while LightGCN takes 720 and 700 epochs in these two datasets respectively. This suggests that our SGL can greatly reduce the training time, meanwhile achieves remarkable improvements.
- Although the training curves on Alibaba-iFashion show different patterns from that on Yelp2018 and Amazon-Book, SGL is still faster to converge, compared with LightGCN. This again validates the benefits of SSL tasks, which offer more supervisions than the supervised learning task.
- Another observation is that the origin of the rapid-decline period of BPR loss is slightly later than the rapid-rising period of Recall. This indicates the existence of a gap between the BPR loss and ranking task. We will conduct in-depth research on this phenomenon in our future work.

4.3.3 ***Robustness to Noisy Interactions.*** We also conduct experiments to check SGL's robustness to noisy interactions. Towards this end, we contaminate the training set by adding a certain proportion of adversarial examples (*i.e.,* 5%, 10%, 15%, 20% negative



(a) Yelp2018 (b) Amazon-Book

Figure 4: Model performance *w.r.t.* noise ratio. The bar represents Recall, while the line represents the percentage of performance degradation.



(a) Yelp2018 (b) Amazon-Book

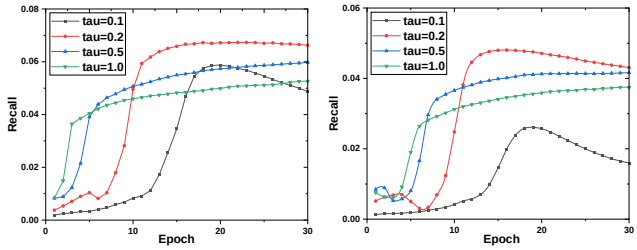
Figure 5: Model performance as adjusting the SSL weight λ_1.

user-item interactions), while keeping the testing set unchanged. Figure 4 shows the results on Yelp2018 and Amazon-Book datasets.

- Adding noise data reduces the performance of SGL and LightGCN, where the curve in Figure 4 indicates the performance degradation of each model. Clearly, the performance degradation of SGL is lower than that of LightGCN; moreover, as the noise ratio increases, the gap between two degradation curves becomes more apparent. This suggests that, by comparing differently augmented views of nodes, SGL is able to figure out useful patterns, especially informative graph structures of nodes, and reduce dependence on certain edges. In a nutshell, SGL offers a different angle to denoise false positive interactions in recommendation.
- Focusing on Amazon-Book, the performance of SGL with 20% additional noisy interactions is still superior to LightGCN with noise-free dataset. This further justifies the superiority and robustness of SGL over LightGCN.
- We find that SGL is more robust on Yelp2018. The possible reason may be that Amazon-Book is much sparser than Yelp2018, and adding noisy data exerts more influence on graph structure of Amazon-Book than that of Yelp2018.

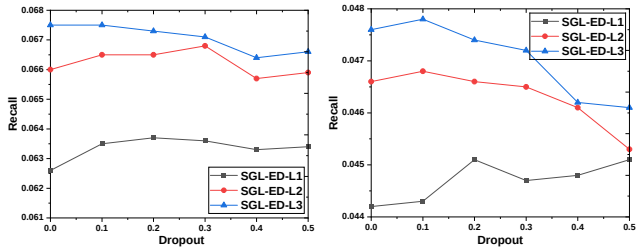
4.4 Study of SGL (RQ3)

As the SSL task plays a pivotal role in SGL, we first investigate the impact of hyper-parameters, *i.e.,* SSL weight, temperature and dropout ratio. We adopt the control variates method, that is, tuning one parameter in the range described in Section 4.1.2 with the other two set as the optimal values. We then study the effect of the SSL task on both user and item side. Lastly, we explore the potential of adopting SGL as a pretraining for existing graph-based recommendation model. For better visualization, we ommit the results on iFashion, which have a similar trend.



(a) Yelp2018 (b) Amazon-Book

Figure 6: Model performance as adjusting τ.



(a) Yelp2018 (b) Amazon-Book

Figure 7: Effect of Dropout Ratios. The suffix in the legend indicates the number of GCN layers.

4.4.1 ***Effect of SSL Weight*** λ_1. Fig. 5 shows the recommendation performance *w.r.t.* Recall under different λ_1 where larger value means higher contribution of $\mathcal{L}_{ssl}$. As can be seen, a proper λ_1 (*e.g.,* 0.1) that balances the two tasks can greatly fasten convergence meanwhile achieving satisfying performance. We also observe overfitting problem when λ_1 is large on Amazon-Book, which suggests us to tune this parameter in fine-grained.

4.4.2 ***Effect of Temperature*** τ. Note that τ controls the scale of $\mathcal{L}_{ssl}$ where a bigger τ leads to bigger SSL loss. Fig. 6 shows the curve of model performance as adjusting the value of τ. From the figure we can observe that: (1) Large τ (*e.g.,* 1.0) will lead to performance drop, which indicates that the similarity function defined in Eq. 11 is insufficient to distinguish positive pair from negative pairs when the value of τ is large. (2) On the contrary, setting τ with value too small (*e.g.,* 0.1) will also hurt the model performance. Therefore, we suggest tuning τ in the range of $0.1 \sim 1.0$.

4.4.3 ***Effect of Dropout*** ρ. Fig. 7 plots the sensitivity analysis of the dropout ratios. We can find that the best performance is achieved when the ratio is in the range of $0.1 \sim 0.3$. This is in line with our intuition that SSL cannot learn sufficient essential characteristics if the two augmented data are not distinct from each other, or losing too much information.

4.4.4 ***Effect of SSL on User and Item Sides.*** As the user-item interaction graph is a heterogeneous graph containing two types of nodes: users and items, we formulate the objective function of SSL as the summation of InfoNCE [12] losses on both user and item sides. Here we conduct ablation studies to verify the influences of user and item nodes. In particular, we create two variants: SGL-ED-user and SGL-ED-item, which separately treat all users and items as negatives in the auxiliary tasks, respectively. Table 5 shows the experimental results. As we can see, the joint optimization on both

sides achieves better performance on both datasets, than that on single sides. It inspires us to make full use of graph structures.

Moreover, we also study the effect of the number of negative samples in the auxiliary task. Two variants are considered: (1) SGL-ED-batch, which differentiates node types and treat users and items in a mini-batch as negative views for users and items, separately, and (2) SGL-ED-merge, which treat nodes within a mini-batch as negative, without differentiating the node types. We report the comparison in Table 5 SGL-ED-batch performs better than SGL-ED-merge, which indicates the necessity for distinguishing types of heterogeneous nodes. Moreover, SGL-ED-batch is on a par with SGL-ED that treats the whole spaces of users and items as negative. It suggests that training the SSL task in mini-batching is an efficient alternative.

4.4.5 ***Effect of Pre-training.*** The foregoing experiments have shown the effectiveness of SGL, where the main supervised task and the self-supervised task are jointly optimized. Here we would like to answer the question: Can the recommendation performance benefit from the pre-trained model? Towards this goal, we first pre-train the self-supervised task to obtain the model parameters, use them to initialize LightGN, and then fine the model via optimizing the main task. We term this variant trained as SGL-pre and show the comparison with SGL in Table 5.

Clearly, although SGL-pre performs worse than SGL-ED on both datasets, the results of SGL-pre are still better than that of LightGCN (*cf.* Table 3). Our self-supervised task is able to offer a better initialization for LightGCN, which is consistent to the observation in previous studies [5]. However, the better performance of joint training admits that the representations in the main and auxiliary tasks are mutually enhanced with each other.

5 RELATED WORK

Our proposed method involves two tasks: graph-based recommendation and self-supervised learning. In this section, we separately review the related work and discuss their relations to our work.

5.1 Graph-based Recommendation

Previous studies on graph-based recommendation can be categorized into: *model-level* and *graph-level* methods, regarding their focus. The model-level methods focus on model design for mining the user-item graph. The research attention has been evolved from the random walk that encodes the graph structure as transition probabilities [2], to GCN that propagates user and item embeddings over the graph [17, 36, 44, 48]. Recently, attention mechanism is introduced into GCN-based recommendation models [43], which learns to weigh the neighbors so as to capture the more informative user-item interactions. A surge of attention has also been dedicated to graph-level methods, which enriches the user-item graph by accounting for side information apart from user-item interactions, which ranges from user social relations [1], item co-occurrence [2], to user and item attributes [26]. Recently, Knowledge Graph (KG) is also unified with the user-item graph, which enables considering the detailed types of linkage between items [4, 42, 43].

Despite the tremendous efforts devoted on these methods, all the existing work adopts the paradigm of supervised learning for model training. This work is in an orthogonal direction, which explores self-supervised learning, opening up a new research line of graph-based recommendation.

Table 5: The comparison of different SSL variants

Dataset	Yelp2018		Amazon-Book	
Method	Recall	NDCG	Recall	NDCG
SGL-ED-user	0.0640	0.0523	0.0445	0.0348
SGL-ED-item	0.0651	0.0535	0.0442	0.0349
SGL-ED-batch	0.0670	0.0549	0.0472	0.0374
SGL-ED-merge	0.0671	0.0547	0.0464	0.0368
SGL-pre	0.0653	0.0533	0.0429	0.0333
SGL-ED	**0.0675**	**0.0555**	**0.0478**	**0.0379**

5.2 Self-supervised Learning

Studies on self-supervised learning can be roughly categorized into two branches: *generative models* [8, 29, 37] and *contrastive models* [5, 9, 15, 38]. Auto-encoding is the most popular generative model which learns to reconstruct the input data, where noises can be intentionally added to enhance model robustness [8]. Contrastive models learn to compare through a Noise Contrastive Estimation (NCE) objective, which can be in either global-local contrast [20, 38] or global-global contrast manner [5, 15]. The former focuses on modeling the relationship between the local part of a sample and its global context representation, *e.g.,* stripes to cats in CV [28].While the latter directly performs comparison between different samples, which typically requires multiple different views of samples [5, 15, 35]. There are both pros and cons of the generative model and contrastive model, this work adopts the contrastive model to avoid additional model parameters.

SSL has also been applied on graph data. For instance, InfoGraph [33] and DGI [40] learns node representations according to mutual information between a node and the local structure. In addition, Hu *et al.* [21] extend the idea to learn GCN for graph representation. Furthermore, Kaveh *et al.* [14] adopt the contrastive model for learning both node and graph representation, which contrasts node representations from one view with graph representation of another view. Besides, GCC [31] leverages instance discrimination as the pretext task for graph structural information pre-training. These studies are focused on general graphs which cannot be directly used to the user-item graph. Moreover, none of the existing work considers augmentation operators from the perspective of embedding.

To the best of our knowledge, very limited work exists in combining SSL with recommendation to date. A very recent one is S^3-Rec [49] for sequential recommendation which utilizes the mutual information maximization principle to learn the correlations among attribute, item, subsequence, and sequence. Another attempt [47] also adopts a multi-task framework with SSL. However, it differs from our work in: (1) [47] uses two-tower DNN as encoder, while our work sheds lights on graph-based recommendation, and devised four augmentation operators from two dimensions: ID embeddings and graph structure. (2) [47] utilizes categorical metadata features as model input, while our work considers a more general collaborative filtering setting with only ID as input.

6 CONCLUSION AND FUTURE WORK

In this work, we recognized the limitations of graph-based recommendation under general supervised learning paradigm and explored the potential of SSL to solve the limitations. In particular, we proposed a model-agnostic framework SGL to supplement the supervised recommendation task with self-supervised learning on user-item graph. From the embedding matrix and graph structure of the GCN-based models, we devised four types of data augmentation from different aspects to construct the auxiliary contrastive task. We demonstrated SGL on LightGCN and conducted extensive experiments on three benchmark datasets, justifying the advantages of our proposal regarding long-tail recommendation, training convergence and robustness against noisy interactions.

This work represents an initial attempt to exploit self-supervised learning for recommendation and opens up new research possibilities. In future work, we would like to make SSL more entrenched with recommendation tasks. Going beyond the stochastic selections on embeddings and graph structure, we plan to explore new perspectives, such as counterfactual learning to identify influential data points, to create more powerful data augmentations. Moreover, we will focus on pre-training and fine-tuning in recommendation — that is, to pre-train a model which captures universal and transferable patterns of users across multiple domains or datasets, and fine-tune it on upcoming domains or datasets. Another promising direction is fulfilling the potential of SSL to address the long-tail issue. We hope the development of SGL is beneficial for improving the generalization and transferability of recommender models.

REFERENCES

[1] Hakan Bagci and Pinar Karagoz. 2016. Context-Aware Friend Recommendation for Location Based Social Networks using Random Walk. In *WWW*. 531–536.

[2] Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*. 895–904.

[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *WWW*. 1341–1350.

[4] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. 151–161.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR* abs/2002.05709 (2020).

[6] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. In *SIGKDD*. 2662–2670.

[7] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. 2009. Power-Law Distributions in Empirical Data. *SIAM* 51, 4 (2009), 661–703.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[9] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, Vol. 70. 1263–1272.

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, Vol. 9. 249–256.

[12] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, Vol. 9. 297–304.

[13] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML*. 3451–3461.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[16] Ruining He and Julian J. McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[18] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *TKDE* 30, 12 (2018), 2354–2366.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[20] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.

[23] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[25] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning Representations for Automatic Colorization. In *ECCV*, Vol. 9908. 577–593.

[26] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *CIKM*. 539–548.

[27] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. 689–698.

[28] Ali Lotfi-Rezaabad and Sriram Vishwanath. 2020. Learning Representations by Maximizing Mutual Information in Variational Autoencoders. In *ISIT*. 2729–2734.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[30] Stasa Milojevic. 2010. Power law distributions in information science: Making the case for logarithmic binning. *J. Assoc. Inf. Sci. Technol.* 61, 12 (2010), 2417–2425.

[31] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[33] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

[34] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and Mitigating Degree-Related Biases in Graph Convolutional Networks. In *CIKM*.

[35] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive Multiview Coding. *CoRR* abs/1906.05849 (2019).

[36] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* abs/1706.02263 (2017).

[37] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional Image Generation with PixelCNN Decoders. In *NIPS*. 4790–4798.

[38] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018).

[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[40] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.

[41] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *NeurIPS*. 4957–4966.

[42] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *WWW*. 3307–3313.

[43] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD*. 950–958.

[44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[45] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. 2018. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *CVPR*. 3733–3742.
[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
[47] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix X. Yu, Aditya Krishna Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi Kang, and Evan Ettinger. 2020. Self-supervised Learning for Deep Models in Recommendations. *CoRR* abs/2007.12865 (2020).
[48] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. 974–983.
[49] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. Sˆ3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *CIKM*.