A Negative Sampling-Based Service Recommendation Method

Ziming Xie[1], Buqing Cao[1](✉), Xinwen Liyan[2], Bing Tang[1], Yueying Qing[1], Xiang Xie[1], and Siyuan Wang[1]

[1] School of Computer Science and Engineering and Hunan Key Laboratory of Service Computing and New Software Service Technology, Hunan University of Science and Technology, Xiangtan, Hunan, China
buqingcao@gmail.com

[2] School of Software and Big Data, Changzhou College of Information Technology, Changzhou, Jiangsu, China

Abstract. As the number of services has increased dramatically in recent years, it has become a challenge to provide users with high quality services. At present, the collaborative filtering technology based on graph neural network has achieved good results in service recommendation. However, these methods ignore the influence of negative sample neighbor nodes. Therefore, we propose a service recommendation method based on negative sampling, which uses the information of neighbor nodes of non-interacted services to generate negative samples to improve the recommendation accuracy. In this method, firstly, we construct the dataset for training based on the interaction records between users and services; second, we use the interpolation mixing technique to pollute the negative samples, and generate synthetic hard negative samples by fusing the negative samples selected by the hard negative selection strategy through the pooling operation; then, the samples are used in GNN-based recommender systems to obtain more accurate user and service embedding representations; next, calculate the inner product score embedded by the user and the service, and recommend high inner product score services to user. Finally, experiments were conducted on three real datasets, Shopify, Programmable, and 360Apps, and the experimental results demonstrate that the proposed method can achieve better recommendation results such as average relative increase of 30% for LightGCN and 8% for NGCF in terms of NDCG@20.

Keywords: Service recommendation · Negative sampling · Graph neural networks

1 Introduction

With the rapid development of technologies such as the internet of things and cloud computing, the number and types of services continue to increase, bringing convenience and information overload to users. It is difficult for users to select the desirable services for them. Therefore, how to select the satisfying services from numerous services and make personalized service recommendations has become a major challenge.

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 3–19, 2022.
https://doi.org/10.1007/978-3-031-24383-7_1

According to our survey, collaborative filtering-based recommender systems perform better in addressing the above challenges [1]. It recommends items to users according to the preferences of groups with common interests. At present, model-based collaborative filtering is the most popular, and it predicts users' ratings of non-interacted items from their interaction data and recommend highly-rated items to users. For example, collaborative filtering based on matrix factorization [2] is a typical method of model-based collaborative filtering, which decomposes the user-item interaction matrix into a user matrix and an item matrix, and then uses the product of the user matrix and the item matrix as the user's rating for the item. Moreover, collaborative filtering based on neural networks [3], which learns a score function from hidden feedback information through neural networks. In order to provide more accurate recommendations to users, some researchers consider modeling the user-item graph structure and utilize the idea of recursively aggregating neighbor information in GNNs to obtain the final embedding of the target node. Currently, NGCF [4] and LightGCN [5] have achieved better results in this work. In general, GNN-based recommender systems first aggregate and propagate neighborhood information through an embedding aggregation layer and an embedding propagation layer, and then obtain the final embedding of users (items) through pooling operations. Finally, the inner product score is used to predict the user's preference for the item. It is generally believed that users prefer interacted items (positive samples) to non-interacted items (negative samples). Taking the classic BPR loss [6] as an example, for each user and each positive sample, we select one of the items that the user has not interact with as a negative sample.

Negative samples largely affect the performance of GNN-based recommender systems. A good negative sampling method can not only improve the calculation efficiency of the model, but also enhance the training effect of the model. Typically, negative sampling uses a uniform distribution [6, 7]. In order to improve the quality of negative samples, some researchers consider using the user's score on the negative samples during model training [8, 9], and use the negative samples with high scores as negative samples. Others researchers consider using GAN to select negative samples [10, 11], but the model is too complicated, the training effect is not good, and the application scenarios are not widely used. Another researcher considers negative sampling combined with graph structure information. For example, PinSage [12] uses PageRank score to select hard negative examples based on the random walk strategy. MCNs [13] proposes to sample negative samples according to an approximate distribution of positive samples, and accelerates negative sampling by Metropolis-Hastings. However, they only consider negative sampling from the perspective of discrete space, ignoring the process of neighborhood aggregation in GNNs.

Inspired by the work of Huang et al. [14], We consider combining the user-item graph structure with a GNN-based aggregation process to synthesize hard negative samples, and propose a negative sampling-based service recommendation method (NSSR). Specifically, the method uses interpolation mixing technology to pollute the negative samples, and fuses the negative samples selected by the hard negative selection strategy through the pooling operation to generate synthetic hard negative samples, which are used in the GNN-based recommendation system. Thus, a more accu rate embedded representation of users and services can be obtained, and finally services are recommended through the inner product of users and services. In summary, the contributions of this paper are as follows:

- This paper proposes a negative sampling-based service recommendation method. The method utilizes the information of non-interacted service's neighbor nodes to generate hard negative samples, and the samples is used for the training of the recommendation system based on graph neural network, which improve the accuracy of service recommendation.
- We inject the information of the interactive services into the negative samples to make the negative samples closer to the positive samples, and use the pooling operation to fuse the information of the hard negative samples of different layers to synthesize high-quality hard negative samples, and improve the recommendation model's ability to distinguish between positive and negative samples.
- We conduct corresponding experiments on the proposed method on real datasets from Kaggle. The experimental results verify the effectiveness of the service recommendation method proposed in this paper, which can better improve the recommendation result of the GNN-based recommendation system.

The rest of this paper is organized as follows. Section 2 is related work on negative sampling methods. Section 3 is the basic process of GNN-based recommender systems. Section 4 details the steps of the method. Section 5 is experimental evaluation and analysis. Section 6 is the conclusion and outlook of this paper.

2 Related Work

2.1 GNN-Based Recommender System

In recent years, with the rapid increase in the number of services, it is difficult for users to find the services they are interested in from a wide variety of services. Therefore, relevant researchers have explored collaborative filtering-based recommendation, usually assuming that users with a similar history of interactions may have the same preferences for the same services.

At present, GNN-based recommender systems have attracted much attention. It models users and services as a bipartite graph, and exploits graph higher-order connectivity to capture collaborative signals from user and service interactions to obtain better latent representations of users and services. Among them, the neighborhood aggregation module affects the quality of the transmitted neighbor information, which largely determines the accuracy of the recommender system. Therefore, NGCF [4] aggregates neighbor information recursively to capture cooperative signals in higher-order connectivity. Specifically, each node obtains its high-order representation by recursively aggregating its neighbor information. When LightGCN [5] does not retain the target node's own information when updating node information through neighbor aggregation and only uses neighbor information as to its representation. It is also found that feature transformation and non-linear activation operations do not contribute to GNN-based collaborative filtering recommendation.

2.2 Negative Sampling Method

The negative sampling method will greatly affect the training quality of the model. Negative samples with better effect can not only improve the training speed of the model, but also provide more valuable information to the model for better training effect. At present, the negative sampling methods mainly fall into the following categories:

Static Negative Sampling. Items are usually selected as negative examples in non-interacted datasets, and by setting different strategies, the corresponding negative samples can be obtained as training data. For example, Rendle et al. [6] propose RNS (Random Negative Sampling) to select a negative example from the negative sample candidate set through a uniform distribution. The PNS (popularity-based negative sampling) proposed by He et al. [16] uses the number of occurrences of items in the training set as the weight for selection, that is, it prefers items with a high frequency of occurrence as negative samples. However, the static negative sampling method does not change with the training, does not adjust well to the training results and makes it difficult to mine negative samples that are more useful to the model.

Hard Negative Sampling. Static negative sampling cannot be adjusted according to the results of training, which may result in some negative samples containing useless information, and making it difficult to mine negative samples that are more effective for the model. Therefore, it is suggested that the discriminative ability of the model can be improved if those samples that are misclassified can be utilized. For example: Zhang et al. [8] consider that the negative examples with high prediction scores can improve the quality of model training, so they propose dynamic negative sampling (DSN), which can dynamically extract the highest scoring negative samples in each training. Kalantidis et al. [9] propose the MoCHi model, which does not directly obtain negative samples from the data, and synthesizes the negative samples directly in contrastive learning. However, hard negative sampling only utilizes the semantic information of the samples in the embedding space, and does not utilize the structural information on the user-item graph.

Graph-Based Negative Sampling. The above studies consider negative sampling using the semantic information of samples in the embedding space, but negative sampling techniques based on graph structure have not been widely explored. Hamilton et al. [16] propose a recommendation model (i.e.SamWalker) that utilizes social network information to obtain negative samples through random wandering at target nodes. Ying et al. [12] proposed the PinSage model based on the random wandering strategy, which further leverages the structural information of the graph to select hard negative examples through PageRank scores. Yang et al. [13] propose Markov Chain Monte Carlo Negative Sampling (MCNs), which proposes to sample negative samples according to the approximate distribution of positive samples, and redesign the sample distribution of positive and negative samples according to the structural correlation on the graph.

Adversarial Sampling. In recent years, generative adversarial networks have gradually become a popular neural network algorithm. The key to the adversarial sampling method lies in the adversarial sampler, which is used to mine the data for negative information.

Wang et al. [10] propose IRGAN, which selects negative examples through GAN, and the generator acts as a negative sample, making it impossible for the discriminator to distinguish positive examples from those selected by the generator. Park et al. [11] propose AdvIR, which is based on GAN, with adversarial training in which adversarial perturbations are imposed. However, the adversarial negative sampling model framework is too complex, the training time is too long, and the training effect is unstable.

3 Preliminaries

Now, we introduce the collaborative filtering recommendation model based on graph neural network (GNN).

When given a set of users $U = \{u\}$, item $V = \{v\}$, positive sample $R^+ = \{(u, v^+)|u \in U,\ v^+ \in V\}$, The recommender system predicts the user's preference for unknown items through the user's historical interaction records in the positive sample.

Recent studies have shown that The CF model based on GNN has obvious advantages in solving the above problems [4, 5, 12]. The main idea is to use the high-order connectivity of the graph to capture the high-order neighbor node information of the user (item) target node to generate the final latent representation, and then express its preference degree through the inner product score. That is, $\hat{y} = e_u^T \cdot e_v$. Next, we'll give a brief introduction to aggregation, pooling, and negative sampling optimizations.

Aggregation. Each user and item have an initial embedding vector e_u and e_v. In the propagation process, by aggregating neighbor information, the cooperative signals of neighbors can be captured. For example, the aggregation process of LightGCN is:

$$e_u^{(l+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^l,\ e_i^{(l+1)} = \sum_{i \in N_i} \frac{1}{\sqrt{|N_u||N_i|}} e_u^l \tag{1}$$

where e_u^l, e_i^l represent the embeddings of user u and item i in the l-th layer of GNN, respectively. $\frac{1}{\sqrt{|N_u||N_i|}}$ is a symmetric normalization term, and $|N_u|, |N_i|$ represent the number of first hop neighbors of user u and item i.

Pooling. GNNs for recommendation are different from GNNs [17, 18] for node classification, which directly use the representation of the last layer, and the former adopt a pooling operation to generate the final representations of users and items. According to [19], adopting a pooling operation helps avoid transition smoothing and determines the degree of validity of node subgraph information.

Specifically, for an L-layer GNN, the pooling operation is used to generate the final representations of users and items. For example, LightGCN uses a sum-based pooling operation:

$$e_u^* = \sum_{l=0}^{L} \lambda_l e_u^{(l)},\ e_i^* = \sum_{l=1}^{L} \lambda_l e_i^{(l)} \tag{2}$$

NGCF uses a sum-based pooling operation:

$$e_u^* = e_u^{(0)} \cdots e_u^{(l)},\ e_u^* = e_u^{(0)} \cdots e_u^{(l)} \tag{3}$$

Negative Sampling Optimization. The purpose of learning to rank is to rank users' scores according to items. Items that users like are generally considered to have higher scores than others. However, we usually only get implicit feedback information from users. Therefore, we assume that users prefer interacted items to non-interacted items. Since the user interacted items are only a small fraction, the BPR loss is used to represent the learning objective:

$$\max \prod_{v^+, v^- \sim f_n(u)} P_u(v^+ > v^- | \Theta) \tag{4}$$

where v^+ and v^- represent positive and negative samples respectively, $P_u(x > y)$ represents that user u prefers x to y, and Θ is a parameter of the model. $f_n(u)$ is the distribution of negative sampling, and most recommended methods use uniformly distributed negative sampling [4–6, 20].

4 Method

The overall framework of the NSSR model proposed in this paper is shown in Fig. 1, including: (1) data preprocessing; (2) synthetic hard negative samples; (3) graph neural network; (4) service recommendation.

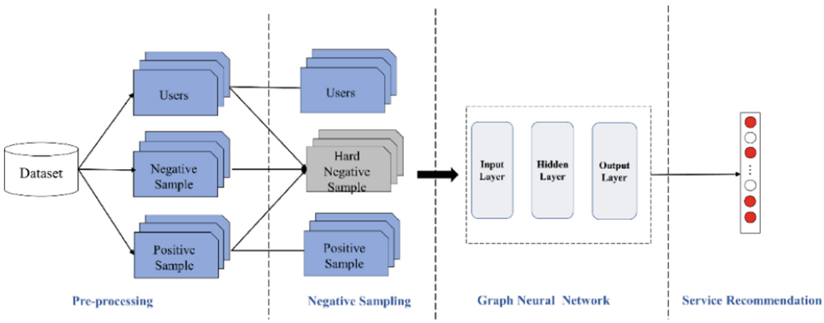

Fig. 1. Service recommendation framework

4.1 Data Preprocessing

This article uses three datasets, Shopify, 360Apps, and ProgrammableWeb respectively, and do the following operations for these three datasets:

Eliminate Invalid Information. In all three datasets, there are some services with incomplete interaction information, and the interaction objects of these services cannot be judged, which will affect the generation of negative samples and the training of neural networks. So, we delete these incomplete interaction records.

Generate the Final Dataset. Since most users only have three or less interaction data, in order to facilitate the training and testing of the neural network, we only retain user data with more than four interaction records. For the Shopify and 360Apps datasets, it still has a lot of interaction data after the above operation. In order to facilitate the training of the model, we randomly extract part of the interaction data according to the proportion of its application categories to generate the final dataset.

Take Raw Negative Samples. Following the convention [15], we adopt the same method to select negative examples from the services that the user has not interacted with to form the candidate set M.

4.2 Synthetic Hard Negative Samples

Positive Mixing. For an $l$layer GNN, each service v has $L+1$ layers embedding vector, and each vector $e_v^{(l)}$ contains the previous l layer information. For the candidate set M.These N negative examples have a total of $N \times (L+1)$ vectors, which constitute a negative sample candidate set$\varepsilon = \{e_{v_i}^{(l)}\}$.

Inspired by the technology of mixup [9, 21], we inject the service information e_{v+} that the user has interacted with into the negative sample vector in ε, which can improves the model's discrimination ability. For each negative sample vector $e_{v_i}^{(l)}$ in the candidate set, the operation of positive mixing is as follows:

$$e'^{(l)}_{v_i} = \alpha^{(l)} e_{v^+}^{(l)} + \left(1 - \alpha^{(l)}\right) e_{v_i}^{(l)}, \alpha \in (0, 1) \tag{5}$$

where $\alpha^{(l)}$ is the l-th layer mixing coefficient uniformly sampled from (0,1), which is used to control the amount of information injected into positive samples and enhance the model's adaptability to different data. And ε' r represents the set of hard negative samples after positive mixing, and $\varepsilon' = \{e'^{(l)}_{v_i}\}$.

Hop Mixing. Positive mixing generates synthetic negative samples v^- and their embeddings e_{v^-} according to the idea of aggregation in GNNs. For the hard negative sample set $\varepsilon' = \{e'^{(l)}_{v_i}\}$obtained in the positive mixture, we extract a hard negative example $e'^{(l)}_{v_i}$ $(1 \le i \le M)$ from each layer $l (0 \le l \le L)$ in the set. For example, when $L = 2$, we can sample $e'^{(0)}_{v_1}$, $e'^{(1)}_{v_2}$and $e'^{(2)}_{v_3}$, where v_1, v_2, v_3 can be the same service or different services. Finally, the hard negative samples sampled in each layer are aggregated through the pooling operation, and get the representation e_{v^-} of the synthetic negative sample v^-. The form is as follow:

$$e_{v^-} = f_{pool}\left(e'^{(0)}_{v_1}, \cdots, e'^{(L)}_{v_n}\right) \tag{6}$$

where f_{pool} is the pooling operation, and its operation is the same as the recommended pooling operation of GNN, $e'^{(l)}_{v_1}$ is the l-th layer embedding vector of v_1.

About how to select a valid $e'^{(l)}_{v_i}$ from each layer of ε'. A recent study on negative sampling for link prediction (MCNS)[13] showed that the expected risk of optimal parameters between expected and empirical losses satisfies:

$$E\left[\|(\theta_T - \theta^*)_u\|^2\right] = \frac{1}{T}\left(\frac{1}{P_d(v|u)} - 1 + \frac{1}{P_n(v|u)} - \frac{1}{K}\right) \tag{7}$$

where $P_d(v|u)$ and $P_n(v|u))$ represent the estimated positive and negative distributions, respectively, and T is the number of user-item node pairs, K is the number of negatives for each user recruiting in the loss. By analyzing the above formula, we can conclude that when the positive distribution is proportional to the negative distribution, the expected risk is only affected by the positive distribution. And the higher the internal score of the user and the item, the more accurate the interaction probability predicted by the recommender system.

Based on the above theory, the problem then becomes how to select negative samples according to the distribution of positive samples. Inspired by the idea that the inner product approximates the positive distribution in the hard negative selection strategy [22, 8], We calculate the inner product of the user vector and the negative sample vector, and select the hard negative sample with the largest inner product. The hard negative sample selection strategy of the l-th layer is of the following form:

$$e'^{(l)}_{v_i} = \text{argmax} f_M(u, l) \cdot e^{(l)}_{v_j} \tag{8}$$

where $e^{(l)}_{v_j} \in \varepsilon^{(l)}$, $f_M(u, l)$ is the mapping function, which represents the user's embedding in the l-th layer, and $\cdot$ is the inner product operation.

The mapping function in Eq. (7) depends on the pooling operation in Eq. (6). At present, the mainstream pooling operations include sum-based and concat-based pooling operations. Therefore, the inner product form of user embedding e_u and negative sample embedding e_{v^-} can be divided into two types:

- Sum-based pooling: $e_u \cdot e_{v^-} = \sum_{l=0}^{L} \lambda_l e_u \cdot e^{(l)}_{v^-}$
- Concat-based pooling: $e_u \cdot e_{v^-} = \sum_{l=0}^{L} \lambda_l e^{(l)}_u \cdot e^{(l)}_{v^-}$

In order to make $f_M(u, l)$ in Eq. (3) consistent with the pooling operation for GNN recommendation, we set $f_M(u, l) = e_u$ for sum-based pooling and $f_M(u, l) = e^{(l)}_u$ for concat-based pooling.

Loss Function. Through the above modeling process, we replace the negative sampling method $f_n(u)$ in Eq. (4) with the NSSR model, and the BPR loss function is updated as:

$$L_{BPR} = -\sum_{\substack{(u, v^+) \in R^+ \\ e_{v^-} \sim f_{WR(u,v^+)}}} \ln\sigma(e_u \cdot e_{v^+} - e_u \cdot e_{v^-}) \tag{9}$$

where $\sigma()$ is the sigmoid activation function, R^+ is the interaction set between the user and the service, and $e_{v^-} \sim f_{WR(u,v^+)}$ means that e_{v^-} is obtained through the NSSR model.

4.3 Graph Neural Network

This paper adopts two graph neural network models, NGCF and LightGCN, for final recommendation prediction. They aggregate the information of neighbors to the target node, which can usually be expressed as:

$$e_u^{(l+1)} = AGG(e_u^{(l)}, \{e_i^{(l)} : i \epsilon N_u\}) \tag{10}$$

where AGG is an aggregation function that aggregates the information of the target node and its neighbors in the l-th layer to obtain a higher-order representation.

This paper uses user feature sampling, positive sample data and strong negative samples generated by the NSSR module as the input data of the graph neural network, and uses the recommendation relationship as the label to train the graph neural network, so that the graph neural network can achieve the ideal recommendation effect. The model uses the inner product of the embedding vectors of users and services as the ranking score of service recommendations. The calculation formula is as follows:

$$\hat{y}_{ui} = e_u^T e_i \tag{11}$$

4.4 Service Recommendation

In the process of recommending services to users, we take the information of users, positive samples and negative samples as input to generate synthetic difficult negative samples, and it is used in the training process based on the GNN recommendation model, so that users can more accurately recommend services of interest. The specific process is as follows:

First, the dataset is preprocessed, according to the convention [15], we select some of the non-interacted services as the negative sample candidate set.

Then, we inject positive sample information into the negative samples, and then fuse the hard negative samples selected by the hard negative selection strategy through the pooling operation to generate synthetic hard negative samples.

Next, we apply the synthesized hard-negative samples to a GNN-based recommendation model which can make model more difficult to distinguish between positive and negative samples and obtain a more accurate final embedding representation of users and services, thereby improving model recommendation quality.

Finally, we obtain user and service embedding representations from model training, and recommend high scoring services to users based on their inner product scores.

5 Experiments

5.1 Dataset Description

The datasets in this paper are the public datasets Shopify app store, Programma-bleWeb, and 360Apps from Kaggle. The Shopify and 360Apps datasets contain user-App interaction data. The Shopify dataset contains 292,029 interaction records, and 360Apps contains 24,631,734 interaction records. To facilitate model training, we first preprocess the data to remove incomplete interaction records and user data with less than 4 interaction records. According to the app category ratio, the data is randomly selected as the final data set. The ProgrammableWeb dataset contains records of different Mashup calling APIs, and we also remove the incomplete interaction records. Finally, we randomly select 30% of the interaction records from the processed dataset as the test set and 70% of the interaction records as the training set. The detailed information is shown in Table 1.

Table 1. Statistics of the dataset

Dataset	User	Item	Interactions	Density
Shopify	5969	2831	44478	0.00263
ProgrammableWeb	6156	1550	12970	0.00136
360Apps	8724	1720	46026	0.00307

5.2 Evaluation Metrics

In this experiment, we use Recall@K, and NDCG@K (default setting K = 20) to evaluate the effectiveness of users' service preference ranking. Where Recall is the proportion of correctly classified services to all the services of this category, and the Normalized Depreciation Cumulative Gain (NDCG), which is used to evaluate the gap between the ranking list of the recommender system and the real user interaction list. The formula and meaning are as follows:

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$DCG = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i + 1)} \tag{13}$$

$$IDCG = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i + 1)} \tag{14}$$

$$NDCG = \frac{DCG}{IDCG} \tag{15}$$

5.3 Baseline Methods

In order to verify the effectiveness of the experimental model, we conduct comparative experiments on the recommendation module and the negative sampling module respectively:

Recommendation Model. To verify the effectiveness of the recommendation module in this paper, we conduct comparative experiments on two GNN-based recommender systems, LightGCN and NGCF.

a) LightGCN[5]: It is a state-of-the-art CF method. LightGCN believes that directly using GCN for collaborative filtering recommendation would make the model too complex, because each node in the user-item graph of CF does not contain any attributes. LightGCN adopts a simple weighted sum aggregator and abandons feature transformation and nonlinear activation to achieve better performance.
b) NGCF[4]: Inspired by graph convolutional networks, NGCF exploits the bipartite graph structure by propagating embeddings. Cooperative signals are explicitly encoded with high-order connectivity. Specifically, each node obtains its high-order representation by recursively aggregating its neighbor information.

Negative Sampling Model. The proposed NSSR negative sampling method is compared with RNS.

a) RNS[6]: Also called uniform negative sampling. During the training process, the probability of each negative sample being selected remains unchanged, that is, one is randomly selected as a negative example from the negative example candidate set.

5.4 Experimental Performance

Table 2. Service recommendation performance comparison

	Shopify		ProgrammableWeb		360Apps	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN + RNS	0.0517	0.0293	0.3977	0.2619	0.4952	0.2916
LightGCN + NSSR	**0.1019**	**0.0514**	**0.4185**	**0.2902**	**0.5163**	**0.3069**
NGCF + RNS	0.0824	0.0424	0.2256	0.1171	0.5082	0.2994
NGCF + NSSR	0.0845	0.0443	0.2523	0.1531	0.4407	0.2611

In this section, we cross comparison the recommendation module and the negative sampling module to analyze the experimental results. From Table 2, Fig. 2 and Fig. 3 we can analyze that:

- LightGCN + NSSR consistently maintains the best performance among all comparison methods. LightGCN has an average improvement of 50% for Recall@20, 30% for NDCG@20.
- In most cases, LightGCN performs better than NGCF, indicating that feature transformation and non-linear activation operations can indeed increase the training burden of collaborative filtering recommendation, which will reduce the accuracy of recommendation.

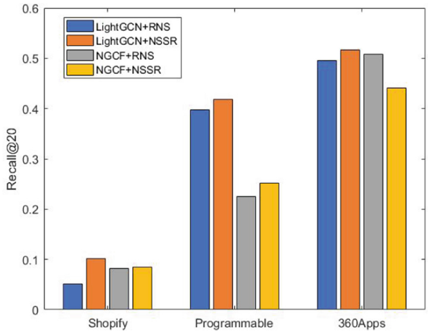

Fig. 2. Comparison of Recall@20 values of different models

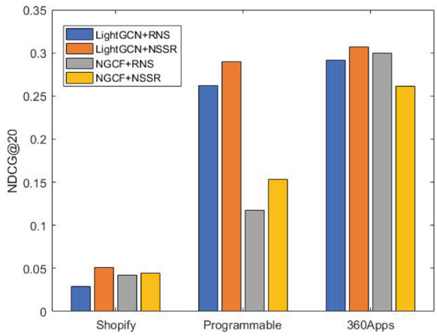

Fig. 3. Comparison of NDCG@20 values of different models

- For both NGCF and LightGCN, NSSR outperforms RNS in almost all cases. We analyze the following reasons: (1) The negative samples are enhanced by the skipmixing technique, which improves the generalization ability of the recommender system. (2) The synthesized hard negative samples are obtained by aggregating multiple hard negative samples from different layers, making the negative samples more informative. The generation of hard negative samples combines semantic information from different samples, thereby providing the model with information-rich gradients.

Table 3. The number of user-App pairs in training and test sets with different sparsity

Shopify	Train	Test
0.26	30282	14196
0.20	22981	10815
0.15	17235	8112
0.10	11490	5408

Table 4. Comparison of service recommendation under different densities in Shopify dataset

	Density	Recall	NDCG
LightGCN + NSSR	0.0026	0.1019	0.0514
	0.0020	0.0903	0.0417
	0.0015	0.0753	0.0341
	0.0010	0.0627	0.0262
LightGCN + RNS	0.0026	0.0517	0.0293
	0.0020	0.0485	0.0259
	0.0015	0.0444	0.0237
	0.0010	0.0357	0.0176

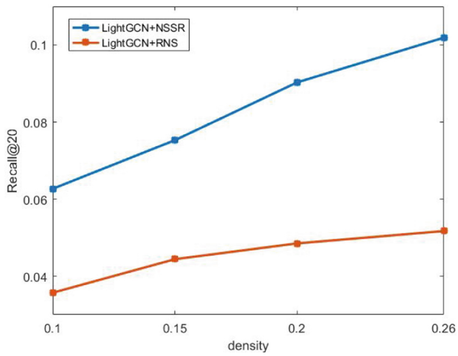

Fig. 4. Recall@20 variation curve under different densities

In order to further explore the performance of different negative sampling methods for different density datasets. We select the above mentioned LightGCN with the best recommendation effect as the recommendation system, and conduct comparative experiments on the Shopify dataset. Without changing the number of users and apps, we vary the sparsity of the dataset by keeping the ratio of its training set to its test set constant and only changing the number of interactions. The number of user-App pairs in the final Shopify dataset for the training and test sets at different densities is set as shown in Table 3.

By comparing Table 4 and Fig. 4 and Fig. 5. We find that both the Recall and NDCG values of NSSR and RNS increase as the dataset density increases, but the Recall and NDCG values of NSSR increase by a larger margin compared to RNS. We analyze that due to the increase of user interaction data, the NSSR model can better capture information from the RNS, enhancing the expression of difficult samples in the positive mixture, while obtaining a better representation of the final negative sample embedding by aggregating richer neighbor information, making it more difficult to distinguish between positive and negative samples, improving the discriminative power of the model and obtaining more accurate recommendations.

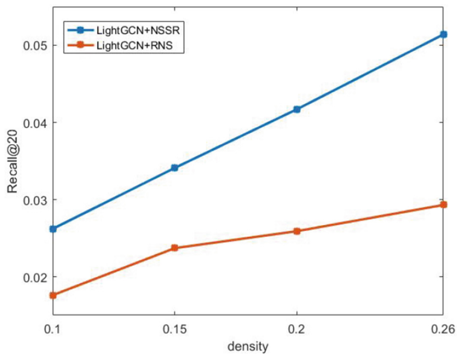

Fig. 5. NDCG@20 variation curve under different densities

5.5 Hyperparameters Analysis

In this section, we will analyze the impact of the three hyperparameters of embedding size, node loss rate, and message loss rate on the performance of recommender systems. The essence of embedding is to replace high dimensional of user and service with a low dimensional vector, the dimension of which is the embedding size (Fig. 6).

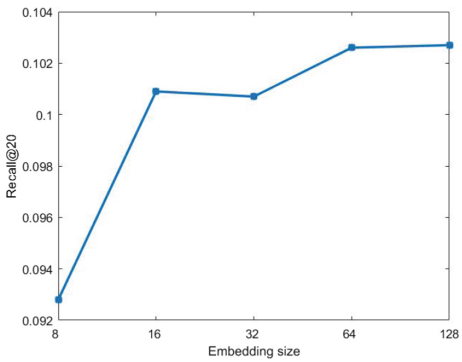

Fig. 6. Recall@20 variation with different embedding sizes

We test the effect of different embedding dimensions of the service on the results of the experiment. As the embedding dimension of the user with the service increases, it effectively improves the results of the experiment. Embedding size increases significantly from 8 to 16, increases slowly from 16 to 64, and flattens out when it is greater than 64. When the embedding dimension is 8, it can be seen that the effect is not ideal, which means that when the embedding dimension is very low, the result will be unsatisfactory because the number of features after embedding of users and services is too small. As the embedding dimension increases, the model becomes more complex, causing the training process to be too slow. When the embedding dimension is 128, it is not much different from the experimental results with the embedding dimension 64, but it is much longer than the time it takes, so we do not conduct relevant experiments on the subsequent embedding dimensions (Fig. 7).

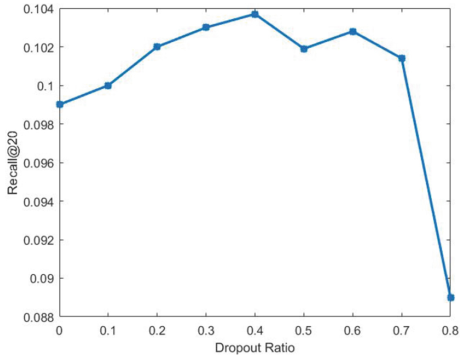

Fig. 7. Variation of Recall@20 in the loss rate of different nodes

We then test the change in Recall@20 for different node loss rates. When the node loss rate changes from 0.0 to 0.4, Recall@20 gradually increases, indicating that the node loss rate within a certain range can make the training effect of the model better. When the node loss rate is greater than 0.6, the training decreases, and when it is greater than 0.7, the effect decreases sharply, indicating that too many nodes loss makes the model obtain too little information, making it difficult for the model to make accurate predictions (Fig. 8).

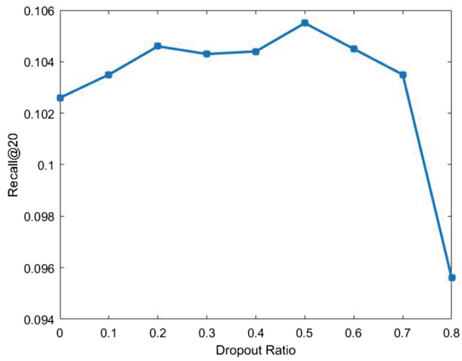

Fig. 8. Variation of Recall@20 with different message loss rates

We also test the variation of Recall@20 for different message loss rates. When the message loss rate changes from 0 to 0.5, the experimental effect is slowly improved, and when the message loss rate is around 0.5, the model accuracy will reach the best. It shows that losing some messages can make model recommendation more effective. When the message loss rate is between 0.5 and 0.7, Reeall@20 decreases slowly, and when the message loss rate is greater than 0.7, Recall@20 decreases significantly, which again shows that too much message loss will greatly affect the model's recommendation performance.

6 Conclusion and Future Work

This paper proposes a negative sampling-based service recommendation method. This method firstly use interpolation mixing technique to inject positive sample information into negative samples, and fuses the negative samples selected by the hard negative selection strategy through the pooling operation to generate synthetic difficult negative samples, which are used in the GNN-based recommendation system to obtain more accurate user and service embedding representation. Finally, according to the inner product score of the user and the service, recommend the service with high inner product score to the user. The experimental results show that the method proposed in this paper can effectively improve the accuracy of service recommendation. In future work, we will consider reducing the number of false negatives, that is, we do not want the model to regard services that users may be interested in as negatives. In future work, we will consider the impact of the selection of negative sampling ratio on the model, and will also consider reducing the number of false negative examples, that is, we do not want the model to regard the services that users may be interested in as negative examples in the future.

Acknowledgment. Our work is supported by the National Natural Science Foundation of China (No. 61873316, 61872139, 61832014, and 61702181), the National Key R&D Program of China (No.2018YFB1402800), Hunan Provincial Natural Science Foundation of China under grant No. 2021JJ30274, and the Educational Commission of Hunan Province of China (No.20B244). Buqing Cao is the corresponding author of this paper.

References

1. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
2. Ma, H., Yang, H., Lyu, M.R., King, I.: Sorec: social recommendation using probabilistic matrix factorization, CIKM2008, pp. 931–940
3. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural Collaborative Filtering, WWW2017, pp. 173–182
4. Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.: Neural Graph Collaborative Filtering, SIGIR2019
5. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, SIGIR2020
6. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback, UAI2009, pp. 452–461
7. He, X., et al.: Fast matrix factorization for online recommendation with implicit feedback, SIGIR2016
8. Zhang, W., Chen, T., Wang, J., Yu, Y.: Optimizing top-n collaborative filtering via dynamic negative item sampling, SIGIR2013, pp. 785–788

9. Kalantidis, Y., Sariyildiz, M.B., Pion, N., Weinzaepfel, P., Larlus, D.: Hard Negative Mixing for Contrastive Learning, NeurIPS2020
10. Wang, J., et al.: IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models, SIGIR2017, pp. 515–524
11. Park, D.H., Chang, Y.: Adversarial Sampling and Training for Semi-Supervised Information Retrieval, WWW2019, pp. 1443–1453
12. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for -scale recommender systems, KDD2018, pp. 974–983
13. Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., Tang, J.: Understanding Negative Sampling in Graph Representation Learning, KDD2020
14. Huang, T., et al.: MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems, KDD2021
15. Huang, J.-T., et al.: Embedding-based Retrieval in Facebook Search, KDD2020
16. Chen, J., Wang, C., Zhou, S., et al.: Samwalker: Social recommendation with informative sampling strategy, WWW2019, pp.228–239
17. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive Representation Learning on Large Graphs, NeurIPS2017, pp.1025–1035
18. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. ICLR2017
19. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.I., Jegelka, S.: Representation Learning on Graphs with Jumping Knowledge Networks. ICML2018
20. Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., Tang, J.: DeepInf: Social Influence Prediction with Deep Learning, KDD2018, pp. 2110–2119
21. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Mixup: Beyond empirical risk minimization, ICLR2018
22. Rendle, S., Freudenthaler, C.: Improving pairwise learning for item recommendation from implicit feedback, WSDM2014, pp.273–282