Towards Automated Negative Sampling in Implicit Recommendation

Fuyuan Lyu*
McGill University
Montreal, Canada
fuyuan.lyu@mail.mcgill.ca

Yaochen Hu
Huawei Noah's Ark Lab
Montreal, Canada
yaochen.hu@huawei.com

Xing Tang
Huawei Noah's Ark Lab
Shenzhen, China
xing.tang@huawei.com

Yingxue Zhang
Huawei Noah's Ark Lab
Montreal, Canada
yingxue.zhang@huawei.com

Ruiming Tang
Huawei Noah's Ark Lab
Shenzhen, China
tangruiming@huawei.com

Xue Liu
McGill University
Montreal, Canada
xueliu@cs.mcgill.ca

ABSTRACT

Negative sampling methods are vital in implicit recommendation models as they allow us to obtain negative instances from massive unlabeled data. Most existing approaches focus on sampling hard negative samples in increasingly complex ways. These studies tend to be orthogonal towards the recommendation model and implicit datasets. However, such an idea contradicts the common belief in automated machine learning that the model and dataset should be matched to achieve better performance. Empirical experiments suggest that the best-performing negative sampler depends on the implicit dataset and the specific recommendation model. Hence, we propose a hypothesis that the negative sampler needs to be aligned with the capacity of the recommendation models as well as the statistics of the datasets to achieve optimal performance. A mismatch between these three would likely results in sub-optimal outcomes. An intuitive idea to address the mismatch problem is to exhaustively or manually search for the best-performing negative sampler given the model and dataset. However, such an approach is computationally expensive and time-consuming. Consequently, the efficient search for the optimal negative sampler remains an unsolved problem. In this work, we propose the AutoSample framework that can automatically and adaptively select the best-performing negative sampler among a set of candidates. Specifically, we propose a loss-to-instance approximation to transform the negative sampler search task into the learning task over a weighted sum, enabling end-to-end training of the model. We also design an adaptive search algorithm to extensively and efficiently explore the search space, which is hard to model directly. A specific initialization approach is also obtained to better utilize the obtained model parameters during the search stage, which is similar to curriculum learning and leads to better performance and less computation resource consumption.

We evaluate the proposed framework on four real-world benchmark datasets over three widely-adopted basic models. Extensive experiments demonstrate the effectiveness and efficiency of our proposed framework. Further ablation studies are also conducted to deepen the understanding of our AutoSample framework from various perspectives.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Collaborative filtering*; • **Computing methodologies** → Discrete space search.

KEYWORDS

Recommendation, Negative Sampling, Automated Machine Learning

1 INTRODUCTION

Collaborative Filtering, as the core component for personalized recommender systems, focuses on learning user preference from observed user-item interactions [26, 27]. Nowadays, recommender systems heavily rely on implicit user feedback, such as purchases on E-commerce sites and views on content platforms. Implicit feedback is more accessible than explicit ones (such as rating) yet more challenging to utilize. Most datasets from implicit feedback only contain positive interactions (e.g., clicking an advertisement, finishing watching a video, etc.), while negative feedback is necessary for training implicit recommendation [27]. Therefore various techniques are widely adopted to sample negative instances from the unobserved user item interactions [10, 23, 26, 27, 36]. An example of such an implicit recommender system is visualized in Figure 1, where the recommendation model takes both the positive instances from the implicit dataset and negative instances from the negative sampler.

Existing negative sampling methods often replace the random sampling [27] with selecting hard negative instances from the pool of unobserved ones [5, 26, 36]. Hard negative instances refer to instances that are difficult to distinguish for the recommendation

*This work was done when the author worked as an intern at Huawei Noah's Ark Lab, Canada.

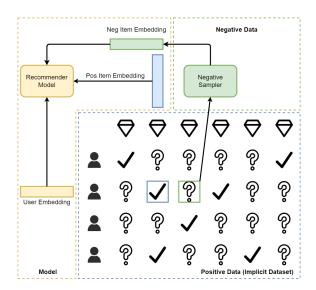**Figure 1: Illustration of An Implicit Recommender System.**

model. Researchers employ various methods as the sampler for choosing hard negative instances, including but not limited to the recommendation model itself [5, 26, 36], the generative adversarial network [24, 29] or graph model [10]. All these methods aim to identify instances that challenge the recommendation model's discriminatory ability.

However, these approaches often disregard the connection between the recommendation model, implicit datasets, and negative sampling methods. We challenge this idea as it contradicts the common belief in standard machine learning tasks, such as computer vision [17] and natural language processing [28], where the model architecture should be typically selected to fit the target dataset to achieve good performance. Unlike these tasks, the implicit recommendation only contains the positive instances of the training dataset, as depicted in Figure 1, while the negative sampler generates the negative instances. Since the positive instances of the dataset are fixed, we hypothesize that the negative sampler should be aligned with the capacity of the recommendation models as well as the statistics of the dataset's positive part to achieve optimal performance. A mismatch among these three components would likely results in sub-optimal outcomes. This hypothesis can be empirically validated by the experimental result in Table 1, where the best-performing negative sampler tends to depend on the dataset and model.

Table 1: The best-performing negative samplers in Table 3.

Model/Dataset	TB2014	TB2015	Alibaba	Amazon
MF [27]	RNS	RNS	PNS	PNS
LightGCN [8]	RNS	MixGCF	RNS	RNS
NGCF [31]	RNS	MixGCF	MixGCF	MixGCF

Here RNS, PNS and MixGCF indicates random sampler [27], popularity-based sampler [23] and MixGCF [10].

An intuitive approach to selecting the suitable negative sampler given the recommendation model and target dataset is to perform an exhaustive and manual search for the best-performing one. However, such a solution would require significant computation resources and time. To mitigate the search cost, we propose an automated negative sampling problem that formulates the selection of negative samplers as a neural architecture search problem, as shown in Figure 2. Directly solving the automated negative sampling problem poses significant challenges, especially in maintaining the end-to-end training flow throughout the search process. To address this challenge and enable end-to-end training, we propose the *instance-to-loss* transformation, which reformulates the problem as a weight-sum of multiple losses generated by different samplers.
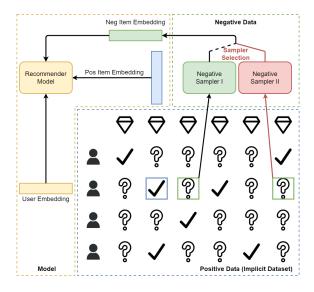

Figure 2: Illustration of Automated Negative Sampling Problem. We highlight the difference with Figure 1 in Red.

To tackle the automated negative sampling problem, we introduce a method named **AutoSample**, which automatically and adaptively selects the sampling strategies for different models and datasets. A gradient-based search algorithm is proposed to explore the search space efficiently. Additionally, inspired by curriculum learning, we propose a specialized retraining scheme to facilitate the model parameters during the search stage, leading to improved performance. We summarize our major contributions as follows:

- This paper first proposes the hypothesis that the performance of negative samplers is dependent on the specific model and dataset, which challenges the existing research approach of developing the negative sampler without considering the model's capability or dataset's statistic.
- This paper introduces the automated negative sampling problem, which can be viewed as a framework that can be combined with existing negative samplers. To ensure end-to-end trainability, we propose the instance-to-loss approximation. Additionally, we present a gradient-based search algorithm and a specialized retraining scheme to efficiently and effectively explore the search space.

• The extensive experiments are conducted on four public datasets and three basic models. The empirical results demonstrate the feasibility and effectiveness of our proposed framework.

2 METHODOLOGY

In this section, we first formulate the implicit recommendation problem in Section 2.1. Then, we formulate the automated negative sampling problem in Section 2.2. Finally, we introduce our method AutoSample in Section 2.3.

2.1 Implicit Recommendation Formulation

Suppose we have a user set $\mathbf{U}$, including M users, and an item set $\mathbf{I}$, including N items. In the implicit recommendation settings, we only have the positive interaction set $\mathbf{D}^p \subseteq \mathbf{U} \times \mathbf{I}$ with each pair $(u, i) \in \mathbf{D}^p$ indicating user u has interacted with item i. To facilitate the training, a negative interaction set $\mathbf{D}^n \subseteq \mathbf{D}_-$ needs to be sampled from the candidate space $\mathbf{D}_- \triangleq \mathbf{U} \times \mathbf{I} - \mathbf{D}^p$. An implicit recommendation model generally consists of three components: (i) a scoring function $\mathcal{S}(u, i)$ to model the relevance between user u and item i, (ii) a loss function $\mathcal{L}$ over the positive pair $(u, i) \in \mathbf{D}^p$ and negative pair $(u, j) \in \mathbf{D}^n$, and (iii) negative sampler $\pi(u, k, \mathbf{D}_-)$ to generate the negative interaction set $\mathbf{D}_u^n = \{(u, j) | j \in \mathbf{I} - \mathbf{I}_u\}$ for each user u with cardinality $|\mathbf{D}_u^n| = k$, where $\mathbf{I}_u = \{i | (u, i) \in \mathbf{D}^p\}$ is the set of iteracted items for user u. Eventually $\mathbf{D}^n \triangleq \cup_u \mathbf{D}_u^n$. Since we focus on the same candidate space $\mathbf{D}_-$, we omit $\mathbf{D}_-$ for simplicity as $\pi(u, k)$.

2.1.1 Scoring Function. The scoring function $\mathcal{S}(u, i | W)$ aims to calculate the relevance score between user u and item i, parameterized by W. There exist various scoring functions based on matrix factorization [26, 27], multiple-layer perceptron [9] and graph neural network [8, 31].

2.1.2 Loss Function. For the loss function $\mathcal{L}$, Bayesian Personalized Loss(BPR) [27] is widely adopted to measure the difference between positive and negative items given the sample user u. It can be formulated as follows:

$$\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W) = -\sum_{(u,i) \in \mathbf{D}^p} \sum_{(u,j) \in \mathbf{D}_u^n} \ln \sigma \left(\mathcal{S}(u, i | W) - \mathcal{S}(u, j | W) \right),$$
(1)

where $\sigma(x) = \frac{1}{1+e^{-x}}$ indicates the sigmoid function. The overall idea for the above equation is to maximize the distance between positive and negative items given the user and scoring function.

2.1.3 Negative Sampler. The negative sampler $\pi(u, k)$ aims to select k negative samples $\mathbf{D}_u^n$ for user u. We consider samplers that draw k samples with the same distribution independently, so we only focus on the sampling process for one negative pair $(u, j) \in \mathbf{U} \times \mathbf{I} - \mathbf{D}_p$. To select one negative sample pair (u, j), the user u is first determined. Here we usually take u for one positive pair (u, i) to fit the BPR loss [27]. The selection of the negative item j, on the other hand, is based on the negative sampling distribution $\mathbf{p}_u$ over each candidate negative samples $j \in \mathbf{I} - \mathbf{I}_u$ for user u, where

$$\mathbf{p}_u(j) > 0, \quad \sum_{j \in \mathbf{I} - \mathbf{I}_u} \mathbf{p}_u(j) = 1.$$
(2)

Various sampling probability has been proposed to facilitate the training of the implicit recommendation model. The most commonly used probability is the uniform distribution [27], where all candidate negative samples have an equal chance to be selected. However, this method usually suffers from low-quality samples [5, 10, 36]. To solve this, various methods have been proposed. PNS [23] heuristically selects the negative samples based on the items' popularity. Hard negative sampling methods, like DNS [36] or AOBPR [26] explicitly favour the candidate samples with higher $\mathcal{S}(u, j)$, which contains more information. There also exist works like IRGAN [29] and AdvIR [24], which simultaneously train a parameterized function $\mathbf{p}_u$ to minimize the loss in Equation 2, based on GAN. Different choices of $\mathbf{p}_u$ are listed in Table 2.

Hence, the final training objective for implicit recommendation can be summarized as follows:

$$\min_W \mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W),$$
(3)

where $\mathbf{D}^n = \cup_u \mathbf{D}_u^n$ and $\mathbf{D}_u^n$ is generated by the negative sampler $\pi(u, k)$.

2.2 Automated Negative Sampling

In this section, we introduce the proposed automated negative sampling problem. The intuition is to adaptively and automatically find the most suitable negative sampler. Consider T different candidate samplers $\pi_1(u, k), \pi_2(u, k), \ldots, \pi_T(u, k)$. To automatically search for the optimal sampler, we assign a learnable weight $\alpha_t \geq 0$ for each sampler $\pi_t(u, k)$. Derived from Equation 3, the automated negative sampling can be formulated as:

$$\min_{W, \boldsymbol{\alpha}} \quad \mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W),$$
(4)

$$\text{s.t.} \quad \mathbf{D}^n = \cup_u \mathbf{D}_u^n, \quad \mathbf{D}_u^n = \cup_t \mathbf{D}_{u,t}^n, \quad \forall u,$$

$$\mathbf{D}_{u,t}^n \text{ is generated by } \pi_t(u, \alpha_t k), \quad \forall u, t$$
(5)

$$\sum_{t=1}^T \alpha_t = 1 \quad \text{and} \quad \alpha_t \geq 0, \forall t.$$

Here $\mathbf{D}_{u,t}^n$ denotes the negative samples generated by sampler $\pi_t(u, \alpha_t k)$ for user u. The search parameter $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_T]$ measures the candidates' contribution to the final negative interaction set $\mathbf{D}^n$. To determine $\boldsymbol{\alpha}$, the naive solution with an exhaustive or manual search of the continuous weight from a large search space is extremely costly. Besides, this formulation is intractable to simultaneously train W and $\boldsymbol{\alpha}$ in an end-to-end manner since the sampling process in Equation 5 cut off the back-propagation path for parameter $\boldsymbol{\alpha}$, losing the useful gradient information.

2.2.1 Instance-level to Loss-level Transformation. To make the recommendation model end-to-end trainable with $\boldsymbol{\alpha}$, we propose a method to transform the loss function to take $\boldsymbol{\alpha}$ from the samplers (instance-level) to the loss (loss-level). Let $\hat{\mathbf{D}}_{u,t}^n$ denotes the negative samples generated by $\pi_t(u, k)$ for $\forall u, t$, and denote $\hat{\mathbf{D}}_t^n = \cup_u \hat{\mathbf{D}}_{u,t}^n$. Then $\mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W)$ is the loss with only sampler $\pi_t(u, k)$. We have

PROPOSITION 1. *Given the set of mutually independent samplers* $\{\pi_t(u,k)|t = 1, 2, \ldots, T\}$, *we have*

$$\mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u,t\}}\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W)$$

$$= \sum_t \alpha_t \mathbb{E}_{\{\pi_t(u,k)|\forall u\}}\mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W) \qquad (6)$$

PROOF. Denote $\delta_{uij} \triangleq \ln \sigma(\mathcal{S}(u, i|W) - \mathcal{S}(u, j|W))$. We have

$$\mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u,t\}}\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W)$$

$$= \mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u,t\}}\left(- \sum_{(u,i)\in\mathbf{D}^p} \sum_{(u,j)\in\mathbf{D}_u^n} \delta_{uij}\right) \quad \text{(by (1))}$$

$$= \mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u,t\}}\left(- \sum_{(u,i)\in\mathbf{D}^p} \sum_t \sum_{(u,j)\in\mathbf{D}_{u,t}^n} \delta_{uij}\right)$$

$$= \sum_t \mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u\}}\left(- \sum_{(u,i)\in\mathbf{D}^p} \sum_{(u,j)\in\mathbf{D}_{u,t}^n} \delta_{uij}\right) \qquad (7)$$

$$= \sum_t \alpha_t \mathbb{E}_{\{\pi_t(u,k)|\forall u\}}\left(- \sum_{(u,i)\in\mathbf{D}^p} \sum_{(u,j)\in\hat{\mathbf{D}}_{u,t}^n} \delta_{uij}\right) \qquad (8)$$

$$= \sum_t \alpha_t \mathbb{E}_{\{\pi_t(u,k)|\forall u\}}\mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W).$$

Equation 7 comes from the linearity of expectation and the independence of the samplers. Equation 8 comes from the fact that $\pi_t(u, k)$ draws k samples independantly for $\forall u, t$. □

Given Proposition 1, we can safely rewrite Problem 4 as

$$\min_{W,\boldsymbol{\alpha}} \quad \sum_t \alpha_t \mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W), \qquad (9)$$

$$\text{s.t.} \quad \hat{\mathbf{D}}_t^n = \cup_u \hat{\mathbf{D}}_{u,t}^n, \forall u,$$

$$\hat{\mathbf{D}}_{u,t}^n \text{ is generated by } \pi_t(u, k), \quad \forall u, t$$

$$\sum_{t=1}^T \alpha_t = 1 \quad \text{and} \quad \alpha_t \geq 0, \forall t.$$

where $\boldsymbol{\alpha}$ can be trained in an end-to-end manner. Proposition 1 guarantees that this substitution is a decent replacement for the original loss.

Table 2: Comparison of Different Negative Samplers.

π	$\mathbf{p}_u(j)$	Category
RNS [27]	$= |\mathbf{I} - \mathbf{I}_u|^{-1}$	Heuristic
PNS [23]	$\propto (\text{pop}_j)^\beta$	Heuristic
AOBPR [26]	$\propto \exp(-\text{grk}(u, j)/\lambda)$	Hard Nega
DNS [36]	$\propto \exp(-\text{lrk}(u, j)/\lambda)$	Hard Nega
IRGAN [29]	learnable	Adversarial Hard Nega
AdvIR [24]	learnable	Adversarial Hard Nega
SRNS [5]	learnable	Hard Nega
MixGCF [10]	learnable	Graph-based Hard Nega

Here *grk()* and *lrk()* refer to global and local ranking, respectively. *pop* stands for popularity.

2.3 AutoSample

In this section, we propose AutoSample, which addresses the automated negative sampling problem discussed in the previous section. Following the convention from the classical neural architecture search problem [17], we introduce the search space, the search algorithm, the search process and the re-training process, respectively.

2.3.1 Search Space. The search space determines which sampler needs to be taken as a potential candidate. In this experiment, we select the candidate based on two criteria: (i) sampling performance and (ii) sampling efficiency. The sampling performance is usually measured by the metrics given the corresponding sampler, whereas the sampling efficiency is important as it may affect the total training time, especially when the dataset is large. The general selection criteria for candidate negative samplers are that we select the top-T best-performing negative samplers as candidates. We discuss the influence of the search space empirically in Section 3.5. Notes that, to improve model efficiency, we only select negative samplers from classic negative samplers with relatively high efficiency, specifically RNS [27], PNS [23], DNS [36] and AOBPR [26].

2.3.2 Search Algorithm. In this section, we illustrate the search algorithm for efficiently exploring the search space. Instead of exhaustively searching for the optimal search parameter, which makes the whole framework not end-to-end differentiable, we approximate the search of negative sampler via the Gumbel-softmax operation [11] in this work. The Gumbel-softmax operation provides a differentiable sampling, which makes the parameters $\boldsymbol{\alpha}$ learnable.

To be specific, suppose architecture parameters $\{\alpha_t|t = 1, 2, \ldots, T\}$ are the search probability over different negative samplers, $\Pi \triangleq \{\pi_t|\forall t\}$ indicates the search space over all possible negative sampling methods. Then a discrete selection z can be drawn via the Gumbel-softmax trick[6] as

$$z = \text{onehot}\left(\underset{\pi_t\in\Pi}{\arg\max}[\log \alpha_t + g_t]\right),$$

$$g_t = -\log(-\log(u_t)), \qquad (10)$$

$$u_t \sim \text{Uniform}(0, 1).$$

The independent and identically distributed (i.i.d) *gumbel noises* g_t disturb the $\log \alpha_t$ terms. Also, they make the $\arg\max$ operation equivalent to drawing a sample from the search weight $\alpha_1, \alpha_2, \cdots \alpha_T$. However, because of the $\arg\max$ operation, this sampling method is non-differentiable. We tackle this problem by straight-through Gumbelsoftmax [6], which leverages a softmax function as a differentiable approximation to the $\arg\max$ operation:

$$p_t = \frac{exp((\log(\alpha_t) + g_t)/\tau)}{\sum_{s=1}^T exp((\log(\alpha_s) + g_s)/\tau)}, \quad \forall t, \qquad (11)$$

where p_t denotes the probability of selecting t-th negative sampler π_t. The temperature τ is the temperature parameter to control the smoothness of the Gumbel-softmax operation. When τ approximates to zero, the Gumbel-softmax operation approximately outputs a one-hot vector. Then we can re-write the loss term in

Equation 9 as

$$\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W) = \sum_t \alpha_t \mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W) \approx \sum_{t=1} p_t \mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W), \quad (12)$$

where $\hat{\mathbf{D}}_t^n = \cup_u \hat{\mathbf{D}}_{u,t}^n$ and $\hat{\mathbf{D}}_{u,t}^n$ is the negative samples generated by $\pi_t(u, k)$. In conclusion, the search algorithm becomes end-to-end differentiable after introducing the Gumbel-softmax operation.

2.3.3 Search Process Optimization. In the above subsections, we formulate and transform the instance-level negative sampler search problem to loss-level. Then we transform the loss-level formulation into a neural architecture search problem and introduce the Gumbel-Softmax trick to make the recommendation model end-to-end differentiable. Now we discuss how we optimize the model during the search process.

In AutoSample, two sets of parameters need to be optimized: the model parameter W contained in scoring function $\mathcal{S}(\cdot, \cdot)$ and the search parameter α. Note that p_t is directly generated by the Gumbel-Softmax operation based on Equation 11. In the naive DARTS [17] setting, two parameters are iteratively trained over different batches. However, in the implicit recommendation, the dataset only contains positive interactions, and it is hard to directly apply such a method. Inspired by the previous work [16, 19], we jointly train both parameters over the same training batch. Such a process can be formulated in Algorithm 1.

Algorithm 1 The Optimization of Search Process

Require: Implicit dataset $\mathbf{D}^p$
Ensure: the well-trained search parameter α^* and model parameter W'
 1: **while** not converged **do**
 2:     Sample a mini-batch of training data $\mathcal{B}^p$ from the implicit dataset $\mathbf{D}^p$
 3:     Generate negative mini-batch $\mathcal{B}^n$ given the current search parameter α and candidate samplers Π
 4:     Calculate the loss $\mathcal{L}(\mathcal{B}^p, \mathcal{B}^n, W)$ given Equation 9
 5:     Update model parameter W by descending gradient $\nabla_W \mathcal{L}(\mathcal{B}^p, \mathcal{B}^n, W)$
 6:     Update search parameter α by descending gradient $\nabla_\alpha \mathcal{L}(\mathcal{B}^p, \mathcal{B}^n, W)$
 7: **end while**

2.3.4 Retraining Process. After obtaining the optimal probability for each sampler α^*, we retrain the model with the best-performing parameter W' as initialization and freeze α^* as the search result for the negative sampler. The idea for keeping the model parameter instead of dropping them as typical neural architecture search methods would is that as α gradually converge to the optimal, the model parameter W is also adaptively trained from easy negative instances to hard ones, similar to curriculum learning [30]. It has also been proven that through careful selection of the initialization parameter, a deep recommender system can achieve better performance than random initialized [20]. The re-training process can be summarized as Algorithm 2.

Algorithm 2 The Re-training Process

Require: Implicit dataset $\mathbf{D}^p$, optimal search parameter α^*, initialization model parameter W'
Ensure: the well-trained model parameter W^*
 1: select the optimal negative sampler π^* given the optimal search parameter α^* and candidate samplers Π
 2: Re-initialize the model with parameter W'
 3: **while** not converged **do**
 4:     Sample a mini-batch of training data $\mathcal{B}^p$ from the implicit dataset $\mathbf{D}^p$
 5:     Generate negative mini-batch $\mathcal{B}^n$ given sampler π^*
 6:     Calculate the loss $\mathcal{L}(\mathcal{B}^p, \mathcal{B}^n, W)$ given Equation 3
 7:     Update model parameter W by descending gradient $\nabla_W \mathcal{L}(\mathcal{B}^p, \mathcal{B}^n, W)$
 8: **end while**

---

3 EXPERIMENT

In this section, to comprehensively evaluate our proposed framework, we design experiments to answer the following research questions:

- **RQ1**: Can the AutoSampler outperform all its candidate samplers and other SOTA baselines?
- **RQ2**: Does the increasing amount of negative instances cause the improvement of AutoSampler?
- **RQ3**: Is our selection algorithm more efficient than exhaustive search?
- **RQ4**: How effective is our two-stage learning algorithm?
- **RQ5**: What kind of samplers does our method perform?

3.1 Experimental Details

3.1.1 Dataset Description. In this paper, we evaluate our method on the following benchmark datasets: Taobao2014[1], Taobao2015[2], Alibaba [33] and Amazon[3]. We randomly split the datasets into training, validation and testing sets by 3:1:1. The statistics and descriptions of these datasets are summarized in Appendix B.1.

3.1.2 Evaluation Metrics. We choose four widely-adopted evaluation metrics: Recall@K(R@K), NDCG@K(N@K), Precision@K(P@K) and Hit Ratio@K(H@K). We choose K=20 as the default setting.

3.1.3 Recommendation Models. All experiments are performed on the three basic recommendation models: Matrix Factorization [27], LightGCN [8] and NGCF [31]. We further compare our methods with six negative sampling baselines: RNS [27], PNS [23], AOBPR [26], DNS [36], SRNS [5] and MixGCF [10]. Notes that we exclude commonly-used adversarial hard negative methods like IRGAN [29] or AdvIR [24] as they are constantly outperformed by SRNS [5] and MixGCF [10]. Details about the recommendation models and negative sampling baselines are shown in Appendix B.2 and B.3. We also present the reproducibility details in Appendix B.5.

---

[1]https://tianchi.aliyun.com/dataset/46
[2]https://tianchi.aliyun.com/dataset/53
[3]http://jmcauley.ucsd.edu/data/amazon/links.html

Table 3: Overall Performance Comparison

	Sampler	Taobao2014					Taobao2015				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0238(2)	0.0247(1)	0.0155(1)	0.2403(1)	1.25	0.0627(2)	0.0412(2)	0.0094(2)	0.1701(2)	2.25
	PNS	0.0220(4)	0.0212(4)	0.0128(4)	0.2013(4)	4	0.0421(6)	0.0248(6)	0.0065(6)	0.1170(6)	6
	DNS	0.0093(6)	0.0093(6)	0.0059(6)	0.1044(6)	6	0.0556(4)	0.0406(4)	0.0082(4)	0.1516(4)	4
	AOBPR	0.0111(5)	0.0115(5)	0.0073(5)	0.1261(5)	5	0.0516(5)	0.0375(5)	0.0077(5)	0.1417(5)	5
	SRNS	0.0221(3)	0.0221(3)	0.0142(3)	0.2207(3)	3	0.0615(3)	0.0417(2)	0.0092(3)	0.1669(3)	2.75
	AutoSample	0.0243*(1)	0.0242(2)	0.0153(2)	0.2355(2)	1.75	0.0688*(1)	0.0449*(1)	0.0103*(1)	0.1854*(1)	1
LightGCN	RNS	0.0327(1)	0.0328(1)	0.0205(2)	0.2959(2)	1.5	0.0696(4)	0.0479(4)	0.0105(4)	0.1873(4)	4
	PNS	0.0267(5)	0.0264(5)	0.0170(5)	0.2535(5)	5	0.0652(5)	0.0454(5)	0.0097(5)	0.1757(5)	5
	DNS	0.0090(7)	0.0091(7)	0.0057(7)	0.1018(7)	6	0.0554(7)	0.0405(7)	0.0081(7)	0.1510(7)	7
	AOBPR	0.0133(6)	0.0144(6)	0.0099(6)	0.1628(6)	6	0.0591(6)	0.0426(6)	0.0088(6)	0.1602(6)	6
	SRNS	0.0316(3)	0.0307(3)	0.0192(3)	0.2847(3)	3	0.0716(3)	0.0489(3)	0.0107(3)	0.1917(3)	3
	MixGCF	0.0288(4)	0.0297(4)	0.0186(4)	0.2755(4)	4	0.0755(1)	0.0504(1)	0.0113(1)	0.2010(1)	1
	AutoSample	0.0321(2)	0.0323(2)	0.0206(1)	0.2982*(1)	1.5	0.0725(2)	0.0493(2)	0.0108(2)	0.1932(2)	2
NGCF	RNS	0.0270(3)	0.0263(2)	0.0173(2)	0.2609(2)	2	0.0572(4)	0.0356(4)	0.0087(3)	0.1557(2)	3
	PNS	0.0268(4)	0.0245(4)	0.0153(4)	0.2378(4)	4	0.0395(7)	0.0225(6)	0.0062(6)	0.1118(5)	5.75
	DNS	0.0088(7)	0.0089(7)	0.0056(7)	0.1025(7)	7	0.0556(5)	0.0407(3)	0.0082(4)	0.1517(3)	3.5
	AOBPR	0.0100(6)	0.0108(6)	0.0076(6)	0.1305(6)	6	0.0452(6)	0.0273(5)	0.0068(5)	0.1260(4)	4.75
	SRNS	0.0227(5)	0.0206(5)	0.0129(5)	0.2023(5)	5	0.0594(3)	0.0390(3)	0.0089(3)	0.1615(3)	3
	MixGCF	0.0274(2)	0.0259(3)	0.0166(3)	0.2501(3)	3	0.0614(2)	0.0429(2)	0.0092(2)	0.1671(2)	2
	AutoSample	0.0308*(1)	0.0293*(1)	0.0187*(1)	0.2747*(1)	1	0.0706*(1)	0.0481*(1)	0.0106*(1)	0.1904*(1)	1

	Sampler	Alibaba					Amazon				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0398(3)	0.0178(3)	0.0024(2)	0.0462(3)	3	0.0310(3)	0.0140(3)	0.0018(2)	0.0355(3)	2.75
	PNS	0.0410(2)	0.0187(2)	0.0024(2)	0.0475(2)	2	0.0317(2)	0.0147(2)	0.0018(2)	0.0358(2)	2
	DNS	0.0201(4)	0.0084(4)	0.0013(4)	0.0253(4)	4	0.0217(4)	0.0073(5)	0.0012(4)	0.0244(4)	4.25
	AOBPR	0.0115(6)	0.0047(6)	0.0007(5)	0.0144(5)	5.5	0.0161(6)	0.0066(6)	0.0009(6)	0.0177(6)	6
	SRNS	0.0124(5)	0.0056(5)	0.0007(5)	0.0142(6)	5.25	0.0198(5)	0.0098(4)	0.0011(5)	0.0222(5)	4.75
	AutoSample	0.0465*(1)	0.0211*(1)	0.0028*(1)	0.0543*(1)	1	0.0354*(1)	0.0155*(1)	0.0020(1)	0.0391*(1)	1
LightGCN	RNS	0.0630(2)	0.0284(3)	0.0038(1)	0.0735(2)	2	0.0416(2)	0.0182(3)	0.0023(2)	0.0465(2)	2
	PNS	0.0587(4)	0.0270(4)	0.0035(4)	0.0688(4)	4	0.0303(5)	0.0141(4)	0.0017(5)	0.0341(5)	4.75
	DNS	0.0234(7)	0.0090(7)	0.0015(7)	0.0290(7)	7	0.0219(6)	0.0072(6)	0.0012(6)	0.0246(6)	6
	AOBPR	0.0336(6)	0.0149(6)	0.0022(6)	0.0423(6)	6	0.0309(4)	0.0131(5)	0.0018(4)	0.0352(4)	4.25
	SRNS	0.0479(5)	0.0222(5)	0.0030(5)	0.0575(5)	5	0.0173(7)	0.0069(7)	0.0010(7)	0.0202(7)	7
	MixGCF	0.0630(2)	0.0305(1)	0.0037(3)	0.0721(3)	2.25	0.0403(3)	0.0189(2)	0.0022(3)	0.0444(3)	2.75
	AutoSample	0.0636*(1)	0.0289(2)	0.0038(1)	0.0745*(1)	1.25	0.0447*(1)	0.0203*(1)	0.0025*(1)	0.0499*(1)	1
NGCF	RNS	0.0464(3)	0.0204(3)	0.0028(2)	0.0548(3)	2.75	0.0269(5)	0.0114(5)	0.0015(5)	0.0304(5)	5
	PNS	0.0369(4)	0.0160(4)	0.0021(4)	0.0422(4)	4	0.0289(4)	0.0128(4)	0.0016(4)	0.0324(4)	4
	DNS	0.0209(5)	0.0090(5)	0.0013(5)	0.0259(5)	5	0.0220(7)	0.0070(7)	0.0012(7)	0.0247(7)	7
	AOBPR	0.0120(7)	0.0051(7)	0.0008(7)	0.0150(7)	7	0.0222(6)	0.0091(6)	0.0013(6)	0.0258(6)	6
	SRNS	0.0188(6)	0.0086(6)	0.0011(6)	0.0220(6)	6	0.0316(3)	0.0132(3)	0.0018(3)	0.0349(3)	3
	MixGCF	0.0473(2)	0.0234(2)	0.0028(2)	0.0550(2)	2	0.0331(1)	0.0148(1)	0.0026(1)	0.0507(1)	1
	AutoSample	0.0485*(1)	0.0246*(1)	0.0029(1)	0.0566*(1)	1	0.0329(2)	0.0137(2)	0.0019(2)	0.0367(2)	2

* denotes improvements over baselines are statistically significant with $p < 0.05$.

3.2 Overall Performance(RQ1)

The overall performance of our AutoSample method and other baselines on three basic models and four benchmark datasets are reported in Table 3. We summarize our observations below.

First, we can observe that our AutoSample consistently outperforms other negative samplers in the experiments. Considering that the four metrics we employed evaluate various aspects of the results, we assess the performance based on the average rank across the four metrics. Our AutoSample ranks first in 9 out of 12 cases,

and even in the remaining three cases, it secures the second rank. These results highlight the superiority of our AutoSample method in terms of model performance.

Furthermore, we can observe that solely targeting hard negative samples does not guarantee constant improvements in all cases. The best-performing baseline methods (excluding our AutoSample) vary from heuristic-sampling methods, like RNS or PNS, to hard negative sampling method, like MixGCF, among all 12 cases. Such an observation validates our hypothesis that the negative sampler

needs to be carefully selected given the model and dataset, which motivates the automated negative sampling problem.

Finally, it is surprising to find out that even though AutoSample limits its search space to classic methods, as stated in Section 2.3.1, it can surpass state-of-the-art sampling methods like SRNS or MixGCF. This finding emphasizes the potential benefits of transforming traditional negative sampling, shown in Equation 3, into an automated negative sampling problem, shown in Equation 4. By introducing more flexibility and adaptability into the negative sampling process, improved performance can be achieved.

3.3 Ablation on The Amount of Negative Samples(RQ2)

In Table 3, AutoSample constantly performs better than other baselines in most cases. But cautious readers may question whether such performance improvement is solely due to the increase in the number of negative instances, as AutoSample introduces T times extra negative instances. This can be observed from Equation 9, where AutoSample utilizes different samplers to independently sample negative instances and merge them into the sample loss function. In this section, we aim to address this question by comparing baselines with the same amount of negative instances.

We experiment on the Taobao2015 dataset using three basic models. As stated in Section 2.3.1, the candidate negative samplers top-T ranked ones in terms of performance. To be specific, for cases where $T = 2$, we adopt $\Pi = [\text{RNS}, \text{DNS}]$ for CF and NGCF model, $\Pi = [\text{RNS}, \text{PNS}]$ for LightGCN model. For cases where $T = 3$, we adopt $\Pi = [\text{RNS}, \text{PNS}, \text{DNS}]$ for all three models.

As observed in Table 4, our AutoSample achieves the best performance in 5 out of 6 cases, which demonstrates that the performance improvement of AutoSample is not solely due to the increase in the number of negative instances, but rather the result of a better alignment between the sampler and other components.

It is also interesting to notice that the influence of increasing the number of negative instances varies. For instance, as T increases, the performance of AOBPR decreases for the CF model while it increases for the LightGCN model. This phenomenon indicates that increasing the number of negative samples does not necessarily lead to improved performance.

3.4 Ablation on the Search Efficiency(RQ3)

In this section, we conduct an ablation study of the search efficiency of AutoSample. We measure the search efficiency by the total training time, which includes the time for obtaining the optimal search parameter α^* and the corresponding model parameter W^*. We compare our method with a grid search that exhaustively runs all candidate negative samplers. The result is shown in Figure 3. We can easily observe that our search algorithm can more efficiently explore the search space than an exhaustive grid search.

3.5 Ablation on Search Space(RQ4)

In this section, we further investigate the influence of search space over the final result. As we stated in Section 2.3.1, we select top-T negative samplers based on their performance as candidate samplers for AutoSample. We aim to validate the effectiveness of this selection criteria empirically.

(a) Alibaba   (b) Amazon

Figure 3: Total Training Time on Alibaba and Amazon.

As we can observe from Table 5, the rank of the corresponding AutoSample positively correlates with the average rank of the search space, denoted as *S.S.A.R.*. Here we calculate the *S.S.A.R.* as $S.S.A.R. = \sum_{i=t}^{T} \text{rank}(\pi_t)/\text{T}$, where $\text{rank}(\pi_t)$ refers to the rank of the negative sampler π_t and is also reported in Table 3. This empirical finding provides further evidence for the feasibility and effectiveness of our selection criteria discussed in the previous section.

3.6 Ablation on Retraining Scheme(RQ4)

In this section, we investigate the influence of the retraining scheme mentioned in Section 2.3.4. We conduct experiments on the Taobao2015 and Amazon datasets using all three basic models. We compare our customized retraining scheme (referred to as *custom*) with random initialization (referred to as *random*) in Table 6. As evident from the results, the customized retraining scheme, which utilizes the best-performing parameters from the search process as model initialization, consistently outperforms random initialization. This ablation study establishes the feasibility of our retraining scheme.

3.7 Case Study(RQ5)

In this section, we perform a case study to examine the changes in search parameter α during the search process. The evolving trends of α at each epoch are reported in Figure 4. As observed, the evolving patterns of α vary across different datasets and models. For cases like NGCF on Alibaba and LightGCN on Amazon, AutoSample exhibits a curriculum learning behaviour, gradually adjusting the negative instances to facilitate better learning. In contrast, for other cases, AutoSample determines the optimal sampler at an early stage, resembling a neural architecture search approach, and trains the model given the searched sampler in the later epochs.

4 RELATED WORK

We discuss how our work is situated in two research topics: (i) negative sampling and (ii) automated machine learning and its application in recommender systems.

Table 4: Ablation over the Number of Negative Samples on Taobao2015 Dataset

	Sampler	T = 2					T = 3				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0629(2)	0.0419(2)	0.0094(2)	0.1706(2)	2	0.0647(2)	0.0437(2)	0.0097(2)	0.1753(2)	2
	PNS	0.0366(5)	0.0214(5)	0.0057(5)	0.1031(5)	5	0.0388(5)	0.0229(5)	0.0060(5)	0.1079(5)	5
	DNS	0.0554(3)	0.0405(3)	0.0081(3)	0.1510(3)	3	0.0555(3)	0.0405(3)	0.0081(3)	0.1511(3)	3
	AOBPR	0.0538(4)	0.0389(4)	0.0079(4)	0.1469(4)	4	0.0525(4)	0.0387(4)	0.0078(4)	0.1445(4)	4
	AutoSample	0.0688*(1)	0.0449*(1)	0.0103*(1)	0.1854*(1)	1	0.0663*(1)	0.0455*(1)	0.0099(1)	0.1793*(1)	1
LightGCN	RNS	0.0692(2)	0.0478(2)	0.0103(2)	0.1856(2)	2	0.0695(1)	0.0479(1)	0.0104(1)	0.1867(1)	1
	PNS	0.0536(5)	0.0343(5)	0.0082(4)	0.1461(5)	4.75	0.0499(5)	0.0307(5)	0.0077(5)	0.1365(5)	5
	DNS	0.0554(4)	0.0405(4)	0.0081(5)	0.1509(4)	4.25	0.0553(4)	0.0405(4)	0.0081(4)	0.1509(4)	4
	AOBPR	0.0643(3)	0.0449(3)	0.0095(3)	0.1736(3)	3	0.0661(2)	0.0456(2)	0.0098(2)	0.1780(2)	2
	AutoSample	0.0725*(1)	0.0493*(1)	0.0108*(1)	0.1932*(1)	1	0.0644(3)	0.0449(3)	0.0095(3)	0.1732(3)	3
NGCF	RNS	0.0573(2)	0.0362(3)	0.0086(2)	0.1555(2)	2.25	0.0571(2)	0.0360(4)	0.0087(2)	0.1554(2)	2.5
	PNS	0.0399(5)	0.0229(5)	0.0063(5)	0.1115(5)	5	0.0427(5)	0.0248(5)	0.0067(5)	0.1188(5)	5
	DNS	0.0553(3)	0.0405(2)	0.0081(4)	0.1506(3)	3	0.0554(4)	0.0406(2)	0.0081(4)	0.1512(3)	3.25
	AOBPR	0.0545(4)	0.0350(4)	0.0082(3)	0.1494(4)	3.75	0.0548(3)	0.0363(3)	0.0082(3)	0.1502(4)	3.75
	AutoSample	0.0706*(1)	0.0481*(1)	0.0106*(1)	0.1904*(1)	1	0.0638*(1)	0.0437*(1)	0.0094*(1)	0.1725*(1)	1

* denotes statistically significant improvements over baselines with $p < 0.05$.

Table 5: Ablation over Search Space on Taobao2015 Dataset.

	Metric	T=2			T=3
		R+P	R+D	D+AO	R+P+D
MF	R@20	0.0654(3)	0.0688(1)	0.0556(4)	0.0663(2)
	N@20	0.0432(3)	0.0449(2)	0.0406(4)	0.0455(1)
	P@20	0.0099(2)	0.0103(1)	0.0082(4)	0.0099(2)
	H@20	0.1734(3)	0.1854(1)	0.1516(4)	0.1793(2)
	Rank	2.75	1.25	4	1.75
	S.S.A.R.	4.13	3.13	4.5	4.08
LightGCN	R@20	0.0725(1)	0.0554(3)	0.0554(3)	0.0644(2)
	N@20	0.0493(1)	0.0405(3)	0.0405(3)	0.0449(2)
	P@20	0.0108(1)	0.0081(3)	0.0081(3)	0.0095(2)
	H@20	0.1932(1)	0.1511(3)	0.1510(4)	0.1732(2)
	Rank	1	3	3.25	2
	S.S.A.R.	4.25	5.5	6.5	5.33
NGCF	R@20	0.0655(2)	0.0706(1)	0.0550(4)	0.0638(3)
	N@20	0.0437(2)	0.0481(1)	0.0403(4)	0.0437(2)
	P@20	0.0098(2)	0.0106(1)	0.0081(4)	0.0094(3)
	H@20	0.1768(2)	0.1904(1)	0.1503(4)	0.1725(3)
	Rank	2	1	4	2.75
	S.S.A.R.	4.38	3.25	4.13	4.08

Here *R, P, D*, and *AO* stand for RNS, PNS, DNS and AOBPR, respectively. *S.S.A.R.* is an abbreviation for search space average rank.

Table 6: Ablation over Retraining Scheme on Taobao2015 and Amazon Datasets.

	Metric	Taobao2015		Amazon	
		random	*custom*	*random*	*custom*
MF	R@20	0.0556	0.0688	0.0281	0.0354
	N@20	0.0406	0.0449	0.0123	0.0155
	P@20	0.0082	0.0103	0.0016	0.0020
	H@20	0.1516	0.1854	0.0323	0.0391
LightGCN	R@20	0.0696	0.0725	0.0416	0.0447
	N@20	0.0479	0.0493	0.0182	0.0203
	P@20	0.0105	0.0108	0.0023	0.0025
	H@20	0.1873	0.1932	0.0465	0.0499
NGCF	R@20	0.0553	0.0706	0.0269	0.0329
	N@20	0.0405	0.0481	0.0114	0.0137
	P@20	0.0081	0.0106	0.0015	0.0019
	H@20	0.1509	0.1904	0.0304	0.0367

Figure 4: Case Study of α on Alibaba and Amazon datasets.

4.1 Negative Sampling

Negative sampling has been an active topic for both industry and academia. It can mainly be summarized into four classes [32]. Heuristic sampling methods rely on specific heuristics to guide the sampling process, including item popularity [23], uniform sampling [27] or employing Markov chains [3, 33]. Hard negative sampling methods, including DNS [36], AOBPR [26], SRNS [5], NSCaching [37], and ReinforceNS [4], assign higher probabilities to instances with significant prediction score, aiming to enhance model training. Adversarial Sampling methods, including IRGAN [29], AdvIR [24], and

GANRec [35], treat the sampling of negative samples as a generative problem and utilize GAN-based methods to improve model training. Graph-based sampling methods, such as RWS [15], RecNS [34], and MixGCF [10] tend to guide the sampling process depending on the underlying graph topology, typically applied in graph-based models. Additionally, some approaches treat all unobserved data as negative [2, 14]. However, this aspect is beyond the scope of discussion in this work.

Our work builds upon prior studies and can be combined with any negative sampling method. It intends to choose the optimal negative sampling methods depending on models and datasets.

4.2 Automated Machine Learning and its Application in Recommender System

Automated machine learning has emerged as a promising approach to automatically select specific tasks or datasets, eliminating the necessity for human experts [17, 18, 25, 38]. It has substantially advanced various domains, such as computer vision [17, 18], natural language processing [28] and feature representation learning [21]. There are two key aspects to automated machine learning: *selection space* and *selection algorithm*. The *selection space* comprises all potential choices and is usually task-dependent. The *selection algorithm* aims to explore the selection space efficiently and can generally be categorized into three classes: controller-based [38], evolution-based [25] and gradient-based [17, 18].

Automated machine learning has found widespread application in recommender systems [1]. It is widely adopted for tasks such as selecting suitable embedding dimensions [12], identifying informative features [20], determining beneficial feature interactions [16], choosing integration function [13, 19] or designing comprehensive architectures [22].

Our work distinguishes itself from the existing research by addressing the challenge of negative sampling in recommender systems. It represents a distinct direction orthogonal to previous studies, and there is potential for further advancements by combining our approach with existing methodologies.

5 CONCLUSION

In this paper, we first challenge the previous negative sampling schemes which explicitly target the hard negative instances and develop negative samplers orthogonal to the implicit datasets and recommendation models. Inspired by the intuition of neural architecture search, we propose a hypothesis that the negative sampler, which generates the negative instances, should be matched with the positive implicit data and the model. Such a hypothesis is empirically proven as the best-performing negative samples vary as the dataset and model change. To solve such a problem, we formulate the automated negative sampling problem and conduct *instance-to-loss* approximation to make it end-to-end trainable. A method called AutoSample is also introduced to efficiently and effectively solve the problem. Extensive experiment over four benchmark datasets and three recommendation models proves the feasibility of AutoSample. Several ablation studies also investigate different settings in AutoSample. Moreover, we also visualize the search result in AutoSample, suggesting that the AutoSample somehow correlates with curriculum learning in obtaining the model parameters.

REFERENCES

[1] Bo Chen, Xiangyu Zhao, Yejing Wang, Wenqi Fan, Huifeng Guo, and Ruiming Tang. 2022. Automated Machine Learning for Deep Recommender Systems: A Survey. *CoRR* abs/2204.01390 (2022). https://doi.org/10.48550/arXiv.2204.01390 arXiv:2204.01390

[2] Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press, New York, NY, USA, 19–26. https://ojs.aaai.org/index.php/AAAI/article/view/5329

[3] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017*. ACM, Halifax, NS, Canada, 767–776. https://doi.org/10.1145/3097983.3098202

[4] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, Sarit Kraus (Ed.). ijcai.org, Macao, China, 2230–2236. https://doi.org/10.24963/ijcai.2019/309

[5] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*. Curran Associates, virtual. https://proceedings.neurips.cc/paper/2020/hash/0c7119e3a6a2209da6a5b90e5b5b75bd-Abstract.html

[6] Emil Julius Gumbel. 1945. Categorical Reparameterization with Gumbel-Softmax. In *Statistical theory of extreme values and some practical applications: a series of lectures, Vol. 33, US*. US Government Printing Office, Washington, D. C., USA.

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). Curran Associates, Long Beach, CA, USA, 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[8] 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 *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. ACM, Virtual Event, China, 639–648. https://doi.org/10.1145/3397271.3401063

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, Perth, Australia, 173–182. https://doi.org/10.1145/3038912.3052569

[10] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, Virtual Event, Singapore, 665–674. https://doi.org/10.1145/3447548.3467408

[11] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France. https://openreview.net/forum?id=rkE3y85ee

[12] Manas R. Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K. Adams, Pranav Khaitan, Jiahui Liu, and Quoc V. Le. 2020. Neural Input Search for Large Scale Recommendation Models. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, Virtual Event, CA, USA, 2387–2397. https://doi.org/10.1145/3394486.3403288

[13] Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenguo Li, and Xiuqiang He. 2020. AutoFeature: Searching for Feature Interactions and Their Architectures for Click-through Rate Prediction. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, Virtual Event, Ireland, 625–634. https://doi.org/10.1145/3340531.3411912

[14] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed H. Chi, and John R. Anderson. 2019. Efficient Training on Very Large Corpora via Gramian Estimation. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, New Orleans, LA, USA. https://openreview.net/forum?id=Hke20iA9Y7

[15] Jing Li, Feng Xia, Wei Wang, Zhen Chen, Nana Yaw Asabere, and Huizhen Jiang. 2014. ACRec: a co-authorship based random walk model for academic collaboration recommendation. In *23rd International World Wide Web Conference, WWW '14*. ACM, Seoul, Republic of Korea, 1209–1214. https://doi.org/10.1145/2567948.2579034

[16] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, Virtual Event, CA, USA, 2636–2645. https://doi.org/10.1145/3394486.3403314

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, New Orleans, LA, USA. https://openreview.net/forum?id=S1eYHoC5FX

[18] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural Architecture Optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). Curran Associates, Montréal, Canada, 7827–7838. https://proceedings.neurips.cc/paper/2018/hash/933670f1ac8ba969f32989c312faba75-Abstract.html

[19] Fuyuan Lyu, Xing Tang, Huifeng Guo, Ruiming Tang, Xiuqiang He, Rui Zhang, and Xue Liu. 2022. Memorize, Factorize, or be Naive: Learning Optimal Feature Interaction Methods for CTR Prediction. In *38th IEEE International Conference on Data Engineering, ICDE 2022*. IEEE, Kuala Lumpur, Malaysia, 1450–1462. https://doi.org/10.1109/ICDE53745.2022.00113

[20] Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing Feature Set for Click-Through Rate Prediction. In *Proceedings of the ACM Web Conference 2023, WWW 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, Austin, TX, USA, 3386–3395. https://doi.org/10.1145/3543507.3583545

[21] Fuyuan Lyu, Xing Tang, Dugang Liu, Haolun Wu, Chen Ma, Xiuqiang He, and Xue Liu. 2023. Feature Representation Learning for Click-through Rate Prediction: A Review and New Perspectives. *CoRR* abs/2302.02241 (2023). https://doi.org/10.48550/arXiv.2302.02241 arXiv:2302.02241

[22] Ze Meng, Jinnian Zhang, Yumeng Li, Jiancheng Li, Tanchao Zhu, and Lifeng Sun. 2021. A General Method For Automatic Discovery of Powerful Interactions In Click-Through Rate Prediction. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, Virtual Event, Canada, 1298–1307. https://doi.org/10.1145/3404835.3462842

[23] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). Curran Associates, Lake Tahoe, Nevada, United States, 3111–3119. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[24] Dae Hoon Park and Yi Chang. 2019. Adversarial Sampling and Training for Semi-Supervised Information Retrieval. In *The World Wide Web Conference, WWW 2019*. ACM, San Francisco, CA, USA, 1443–1453. https://doi.org/10.1145/3308558.3313416

[25] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka I. Leon-Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-Scale Evolution of Image Classifiers. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, Sydney, NSW, Australia, 2902–2911. http://proceedings.mlr.press/v70/real17a.html

[26] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, February 24-28, 2014*. ACM, New York, NY, USA, 273–282. https://doi.org/10.1145/2556195.2556248

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Jeff A. Bilmes and Andrew Y. Ng (Eds.). AUAI Press, Montreal, QC, Canada, 452–461. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25

[28] David So, Quoc Le, and Chen Liang. 2019. The Evolved Transformer. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, CA, USA, 5877–5886. https://proceedings.mlr.press/v97/so19a.html

[29] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017*. ACM, Shinjuku, Tokyo, Japan, 515–524. https://doi.org/10.1145/3077136.3080786

[30] Xin Wang, Yudong Chen, and Wenwu Zhu. 2022. A Survey on Curriculum Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 9 (2022), 4555–4576. https://doi.org/10.1109/TPAMI.2021.3069908

[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019*. ACM, Paris, France, 165–174. https://doi.org/10.1145/3331184.3331267

[32] Lanling Xu, Jianxun Lian, Wayne Xin Zhao, Ming Gong, Linjun Shou, Daxin Jiang, Xing Xie, and Ji-Rong Wen. 2022. Negative Sampling for Contrastive Representation Learning: A Review. *CoRR* abs/2206.00212 (2022). https://doi.org/10.48550/arXiv.2206.00212 arXiv:2206.00212

[33] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding Negative Sampling in Graph Representation Learning. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, Virtual Event, CA, USA, 1666–1676. https://doi.org/10.1145/3394486.3403218

[34] Zhen Yang, Ming Ding, Xu Zou, Jie Tang, Bin Xu, Chang Zhou, and Hongxia Yang. 2022. Region or Global A Principle for Negative Sampling in Graph-based Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 1-1 (2022), 1–1. https://doi.org/10.1109/TKDE.2022.3155155

[35] Zhi Yang, Jiwei Qin, Chuan Lin, Yanping Chen, Ruizhang Huang, and Yongbin Qin. 2023. GANRec: A negative sampling model with generative adversarial network for recommendation. *Expert Syst. Appl.* 214 (2023), 119155. https://doi.org/10.1016/j.eswa.2022.119155

[36] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13*, Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai (Eds.). ACM, Dublin, Ireland, 785–788. https://doi.org/10.1145/2484028.2484126

[37] Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019. NSCaching: Simple and Efficient Negative Sampling for Knowledge Graph Embedding. In *35th IEEE International Conference on Data Engineering, ICDE 2019*. IEEE, Macao, China, 614–625. https://doi.org/10.1109/ICDE.2019.00061

[38] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France. https://openreview.net/forum?id=r1Ue8Hcxg

A   ETHICAL CONSIDERATIONS

The proposed AutoSample focuses on sampling method in recommendation models. As with other recommendation models, our method may cause filter bubbles or echo chamber phenomena to users of the service. Therefore, when deploying our AutoSample with the recommendation model, it is necessary to combine some existing debiasing techniques to reduce these adverse effects.

B   EXPERIMENTAL SETTINGS

B.1   Dataset Description

In this paper, we evaluate our method on the following benchmark datasets: Taobao2014, Taobao2015, Alibaba and Amazon. The statistics of these datasets are summarized in Table 7. We randomly split the datasets into training, validation and testing sets by 3:1:1. Belows list the detailed description of each dataset:

- Taobao2014[4] contains real user behaviour data from 2014 provided by Alibaba Group. We filtered out user and item nodes with fewer than 10 interactions.
- Taobao2015[5] is a real-world dataset accumulated on Tmall / Taobao and the app Alipay in 2015. We filtered out user and item nodes with fewer than 10 interactions.
- Alibaba dataset [33] is constructed based on the data from another large E-commerce platform, which contains the user's purchase records and items' attribute information. The data are organized as a user-item graph for the recommendation.

---

[4]https://tianchi.aliyun.com/dataset/46
[5]https://tianchi.aliyun.com/dataset/53

- Amazon[6] is a large E-commerce dataset introduced in previous work [10, 33], which contains purchase records and review texts whose time stamps span from May 1996 to July 2014. In the experiments, we take the data from the commonly-used "electronics" category to establish a user-item graph.

Table 7: Dataset Statistics

Datasets	#user	#item	#interactions	Density(‰)
Taobao2014	8844	39103	749438	2.1671
Taobao2015	92605	9842	1332602	1.4621
Alibaba	106042	53591	907470	0.1597
Amazon	192403	63001	1689188	0.1394

B.2 Recommendation Models

To verify the effectiveness of our method, we perform experiments on the following three recommendation models.

- Matrix Factorization [27] is a commonly-used model in implicit recommender systems to learn both the users' and items' embedding. It calculates the relevance score between a user and an item by calculating the inner product of their embeddings.
- LightGCN [8] is a lightweight version of the GCN [7], customized for recommender system. It only contains the neighbourhood aggregation component for collaborative filtering.
- NGCF [31] is a widely-used graph recommendation model. It proposes to integrate the user-item interactions into the embedding process by injecting the collaborative signal into the embedding process in an explicit manner.

B.3 Baselines

We compare our proposed method with various baseline methods as follows. For all sampling-based methods, we sample one negative instance for each positive instance following previous works [29].

- RNS [27], or random sampling, selects negative samples from all candidates with a uniform probability.
- PNS [23], or popularity-based sampling, selects negative samples based on the popularity of each item. We set the hyper-parameter $\beta = 0.75$ following previous work [23].
- AOBPR [26] deliberately selects the hard negative samples with the relevance scores in the recommendation models. It calculates the relevance scores for all unobserved instances and chooses the one with the highest relevance score as the negative instance.
- DNS [36], or dynamic negative sampling, also aims to select the hard negative samples with the relevance scores in the recommendation models. It selects a fixed number of unobserved instances as candidates and chooses the one with the highest relevance score as the negative instance.
- SRNS [5] is a SOTA sampling method that wishes to select hard negative samples. It distinguished the hard negative and false negative instances by the variance of relevance scores and favoured the false negative ones with high variance. And it keeps a designed memory that only stores a few important candidates, which boosts the sampling efficiency.

- MixGCF [10] is a recent method targeting graph-based negative sampling. Instead of sampling negative instances, it synthesizes them by aggregating the neighbours' embeddings. Notes that for MixGCF, we only compare it on LightGCN and NGCF as it is a method targeting graph-based recommendation models instead of general ones.

B.4 Platform

All experiments are conducted on a Linux server with 8 Intel Xeon Gold 6154 cores, 64 GB memory and one Nvidia-V100 GPU with PCIe connections.

B.5 Reproducibility and Hyper-parameters

Our implementation[7] is adapted from the official PyTorch implementation for the MixGCF baseline[8]. For other baseline methods, we reuse the official implementation for MixGCF and SRNS[9] [5]. We re-implement the RNS [27], PNS [23], AOBPR [26] and DNS [36] in our implementation due to either lack of official implementations or the official repositories are supported by early-stage deep learning frameworks.

We also provide the hyper-parameter tuning details for our experiments. For AutoSample, (i) General hyper-parameters: We set the embedding dimension to 64 and batch size to 1024. We set the additional network hyper-parameter introduced by LightGCN and NGCF the same as the previous paper [10]. Following previous work [10], Adam optimizer is adopted. We tune the optimal learning ratio from {3e-3, 1e-3, 3e-4, 1e-4} and l_2 regularization from {1e-2, 1e-3, 1e-4, 0}. (ii) Additional hyper-parameters: For the selection parameter α, we tune its learning ratio from {3e-2, 1e-2, 3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5} and exclude any regularization as we do not want to incorporate prior information. During the re-training process, we reuse the optimal learning ratio and l_2 regularization. For MixGCF, we select its optimal hyperparameter reported in the paper. For SRNS, as it was never evaluated on the four benchmarks in our settings, we select the optimal hyperparameter from the same hyperparameter domain of AutoSample. We will report the exact optimal hyperparameters used in the experiments after the paper is accepted.

---

[6]http://jmcauley.ucsd.edu/data/amazon/links.html

[7]https://anonymous.4open.science/r/AutoSample

[8]https://github.com/huangtinglin/MixGCF

[9]https://github.com/dingjingtao/SRNS/