UFNRec: Utilizing False Negative Samples for Sequential Recommendation

Xiaoyang Liu*§ Chong Liu †§ Pinzheng Wang‡ Rongqin Zheng† Lixin Zhang† Leyu Lin† Zhijun Chen* Liangliang Fu*

Abstract

Sequential recommendation models are primarily optimized to distinguish positive samples from negative ones during training. Thus, negative instances sampled from enormous unlabeled data are essential in learning the evolving user preferences through historical records. Except for randomly sampling negative samples from a uniformly distributed subset, many delicate methods have been proposed to mine negative samples with high quality. However, due to the inherent randomness of negative sampling, false negatives are inevitably collected in model training. Current strategies mainly focus on removing such false negatives, which leads to overlooking potential user interests, lack of recommendation diversity, less model robustness, and suffering from exposure bias. To this end, we propose a novel method that can **U**tilize **F**alse **N**egative samples for sequential **Rec**ommendation (UFNRec), which thoroughly explores the leverage of false negatives. We first devise a simple strategy to extract false negatives from true negatives and directly reverse the labels of false negatives. To avoid extra noise from reversed samples, we restrict false negatives in the output space by an EMA operation and a consistency regularization loss. To the best of our knowledge, this is the first work to utilize false negatives instead of simply removing them for sequential recommendation. Both offline and online experiment results demonstrate that UFNRec can effectively draw information from false negatives and further improve the performance of SOTA models. Recently, we have deployed UFNRec on real-world recommendation servings. The code is available at `https://github.com/UFNRec-code/UFNRec`.

Keywords: Sequential Recommendation; Negative Sampling; Consistency Training

1 Introduction

Recommendation systems play an essential role in online platforms [2, 11] to learn evolving user preferences and fulfill user requirements. Generally, the core of sequential recommendation is to optimize neural models that can distinguish positive samples from negative ones based on users' historical iterations. Many methods have been proposed to capture user interests and recommend items accurately, including RNN-Based methods [5], GNNs structures [9], CNNs frameworks [6], and model variants [8, 1, 10] based on the multi-head self-attention [7]. Though promising, these methods commonly apply items with historical interactions (e.g., clicks) as positive samples and regard randomly selected items as negative samples. However, due to the inherent randomness of negative sampling, false negatives naturally exist in negative samples, which means a selected negative item can be clicked by the user in the future. Since false negatives are potential positive samples, directly involving false negatives during training can affect the recommendation quality. Thus, we explore how to utilize false negatives reasonably in this paper.

However, it is non-trivial to devise an effective method to utilize false negatives in the following two noteworthy aspects: 1) Due to the randomness of negative samples and the lack of reliable supervised signals for false negatives, it is challenging to design a valid criterion to distinguish false negatives from all randomly selected negative ones. Significantly, such a criterion should be instance-level and context-aware. For example, if a negative sample fits the interests of the corresponding user or is highly related to the positive item, this sample can be a false negative sample. 2) Because of the inherent uncertainty of such false negatives, we should carefully design an approach to utilize these samples. On the one hand, simply removing these samples can avoid introducing noise but roughly ignore the significant benefits of such samples. On the other hand, directly involving these samples in the training process can bring additional noise and lead to inferior model performance. Thus, the strategy to utilize false negatives should be thorough enough.

Many studies have paid attention to the influence of false negatives [21, 22]. Early studies [20] rely on extra information, such as item content, to identify possible false negatives and train them with low weights. Recently, SRNS leverages a variance-based sampling strategy to filter out false negatives but roughly removes these samples during training, which overlooks

*OPPO,{liuxiaoyang,justy.chen,fuliangliang}@oppo.com
†Tecent,{nickcliu,leonezheng,lixinzhang,goshawklin}@tencent.com
‡Soochow University,pzwang@stu.suda.edu.cn
§Equal Contribution.

the importance of false negatives in enhancing user experience. Besides, studies in Knowledge Graph (KG) also find that introducing false negatives will lead to inferior model performance. With the observation that the scores of false negatives in KG can be very high, NSCaching [23] directly removes the sample with the largest score in each negative sample set. Though devising effective criteria to identify false negatives, these methods only focus on reducing the risk of involving false negatives while ignoring the potential value of false negatives to improve model performance.

To this end, in this paper, we propose an effective strategy to utilize false negatives for sequential recommendation. With the observation that negative samples with stable larger scores than positive ones tend to be false negatives, we leverage a score-based criterion to filter out false negatives from negative samples without involving extra data. Unlike SRNS [13] that removes false negatives, we utilize these samples to train the model further. To improve label quality and avoid extra noise, we provide both newly defined hard labels and reliably generated soft labels for false negatives. As for hard labels, we directly reverse the labels of these samples and treat them as positive samples during the following training process, which can widen user interests and improve the recommendation diversity. To avoid noise from the above hard labels, we generate a mean teacher from current models based on self-training and regard stable predictions of the teacher model as soft labels for false negatives. Concretely, for false negatives, we adopt a consistency regularization loss on the teacher-student framework to calibrate the predictions of the current model. Experiments on three backbone models and three real-world datasets prove that UFNRec can improve model performance for sequential recommendation over different SOTA models. Moreover, we conduct online experiments to evaluate the effectiveness of UFNRec in online servings. This work mainly includes the following contributions:

- We propose an effective approach for sequential recommendation systems to identify and utilize false negative samples. To our best knowledge, this is the first work to utilize false negative samples to improve sequential recommendation performance.
- We creatively transfer false negatives to positive ones and introduce a consistency regularization method based on self-training to calibrate predictions of false negatives.
- Both online and offline experiments indicate the superiority of our proposed UFNRec over SOTA models. Recently, we have deployed UFNRec on real-world recommendation servings.

2 Related Work

Sequential recommendation aims to learn evolving user preferences based on historical interactions. Early works widely apply Markov Chain (MC) to model sequential interactions, including both the first-order MC [3] and the high-order MC [4]. Recently, many deep learning approaches are proposed to capture user interests and enhance model performance. For instance, GRU4Rec [5] applies an RNN-based method to learn sequential interactions, while Caser [6] utilizes CNN to extract users' short-term preferences. Besides, SASRec [1] introduces a self-attention mechanism [7] to identify relevant items from users' interaction history and recommend the next item. Moreover, combining attention mechanisms with GNNs [9] proves to be another alternative way for the SR task. Though promising, these methods mainly regard items randomly selected from a specific set, e.g., the training dataset or a mini-batch samples, as negative samples, ignoring that the quality of negative samples can affect the model performance. Recently, MNS [12] mixes uniformly and in-batch negative samples to alleviate the selection bias. CBNS [14] focuses on cross batch negative sampling strategy instead of only sampling negative data in an in-batch way.

However, these methods mainly concentrate on effectively mining negative samples while overlooking the existence of false negative samples [22]. Few studies notice the influence of false negative samples for the sequential recommendation. [20] identifies possible false negative samples by additional information and then reduces the training weights of these samples. [21] interprets the complicated and varied reasons for users' inaction and emphasizes that inaction information brings benefits to recommendation systems. In the domain of knowledge graph, NSCaching [23] finds that the scores of false negative samples can be very high and thus removes the sample with the largest score in each negative sample set. For recommendation tasks, SRNS [13] recently proposes a variance-based strategy to filter out false negative samples and removes these samples during training. Though achieving strong performance, these methods only focus on how to distinguish false negative samples from all negative samples and then reduce the risk of introducing these samples. Different from their studies, we address the utilization of false negative samples, which can improve model robustness and recommendation diversity.

3 The UFNRec Model

Figure 1 shows the overall architecture of our model. Before fully explaining UFNRec, we introduce some basic notations to describe the sequential recommendation task. Let $\mathcal{V} = (v_1, v_2, ..., v_{|\mathcal{V}|})$ and $\mathcal{U} = (u_1, u_2, ..., u_{|\mathcal{U}|})$

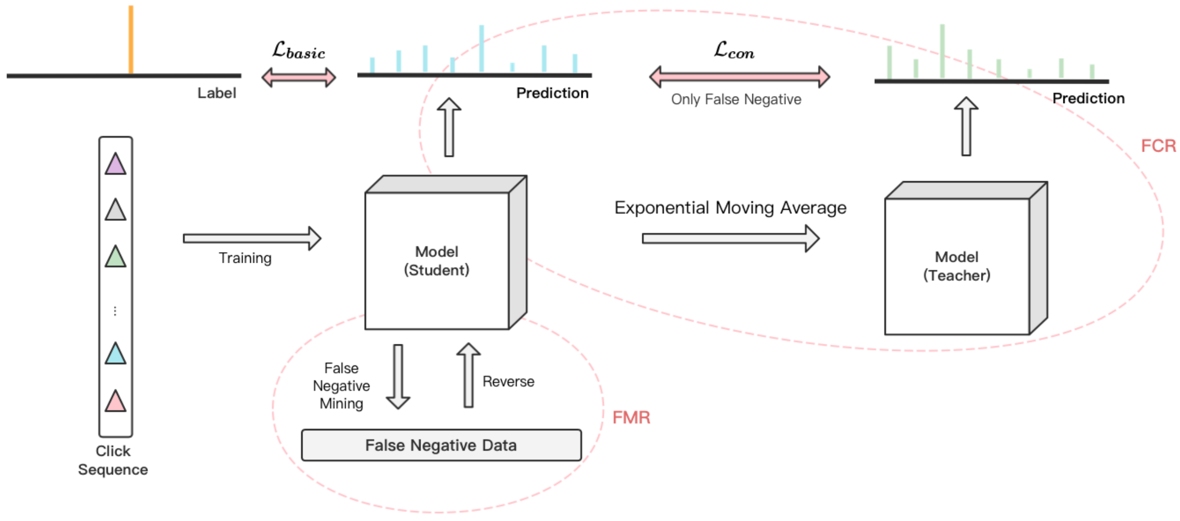

Figure 1: Model structure of UFNRec. It filters out false negative samples and reverses their labels to train the model. In addition, a teacher model is generated from current student models by EMA, and a consistency loss is introduced to regularize outputs from the teacher-student framework.

respectively denote a group of items and users. For a given user $u \in \mathcal{U}$, the historical interaction sequence is denoted as $s_u = (v_1^{(u)}, v_2^{(u)}, ..., v_t^{(u)}, ..., v_{|s_u|}^{(u)})$, where $v_t^{(u)} \in \mathcal{V}$ is the item interacted with user u at time step t and $|s_u|$ denotes the length of s_u. The goal of sequential recommendation is to predict the probability of all alternative items based on the historical sequence s_u and then recommend the most likely item that the user u will interact with at time step $|s_u|+1$. This prediction task can be formulated as $P(v_{|s_u|+1}^{(u)} = v|s_u)$.

3.1 Backbone Model Since our proposed UFNRec method does not depend on structural information, we choose three widely used models (i.e., SASRec [1], BERT4Rec [8] and SSE-PT [18]) as backbones. Generally, user representation $\boldsymbol{s_u} = f(s_u)$ can be obtained at each time step t, where $f(\cdot)$ indicates the model encoder. To learn the correlation between users and items in the sequential recommendation, a similarity function, e.g., inner product, is used to measure the distance between item representation and user representation. Then, as shown in Eq. 3.1, a binary cross entropy loss function $l_{u,t}(\boldsymbol{v_{t+1}}) = l(\boldsymbol{v_{t+1}}|\boldsymbol{s_{u,t}})$ can be inferred for the user representation $\boldsymbol{s_{u,t}}$ of user u at time step t.

$$\begin{aligned} l_{u,t}(\boldsymbol{v_{t+1}}) = &-ylog(\sigma(\boldsymbol{s_{u,t}}\boldsymbol{v_{t+1}})) \\ &-(1-y)log(1-\sigma(\boldsymbol{s_{u,t}}\boldsymbol{v_{t+1}})) \end{aligned} \tag{3.1}$$

Here, $\sigma(\boldsymbol{s_{u,t}}\boldsymbol{v_{t+1}})$ is the prediction result with $\sigma(x) = 1/(1+e^{-x})$, and $y \in \{0,1\}$ is the label. Commonly, for each positive sample, n negative items are randomly selected from dataset $\mathcal{V}$ to obtain a set $\mathcal{N}$ with $|\mathcal{N}| = n$. Then, a basic loss is utilized to measure the prediction accuracy for the positive item $\boldsymbol{v_{t+1}^+}$ and n negative items $\boldsymbol{v_{t+1}^-}$ from the set $\mathcal{N}$:

$$\mathcal{L}_{basic}(\boldsymbol{s_{u,t}};\omega) = l_{u,t}(\boldsymbol{v_{t+1}^+};\omega) + \sum_{\boldsymbol{v_{t+1}^-}\in\mathcal{N}} l_{u,t}(\boldsymbol{v_{t+1}^-};\omega) \tag{3.2}$$

3.2 False Negative Mining and Reversing (FMR) As mentioned above, the basic loss of SR models relies on both positive and negative samples. Due to the randomness of negative samples, the quality of set $\mathcal{N}$ is doubtful, and the existence of false negatives is inevitable. Without discriminating false negatives, the model will train with both true negatives and false negatives, making the model less robust. To address this problem, we design an efficient strategy to filter out false negatives and reverse their labels to train the model.

False Negative Mining. Motivated by the observation that false negatives always have larger scores than true negatives [23], we identify false negatives according to their prediction results. Before applying our strategy, we first warm up the model with randomly sampling until the basic loss converges and then start to mine false negatives. Concretely, during each epoch, we record negative samples (i.e., set $\mathcal{N}_{rec}$) with larger prediction results than positive ones (i.e., $\sigma(\boldsymbol{s_{u,t}}\boldsymbol{v_{t+1}})$). Then, we retrain them with other randomly sampled negative items (i.e., set $\mathcal{N}_{ran}$) in the next epoch following Eq. 3.2 with $v_{t+1}^- \in \mathcal{N}_{ran} \cup \mathcal{N}_{rec}$ and $|\mathcal{N}_{ran}| = n - |\mathcal{N}_{rec}|$.

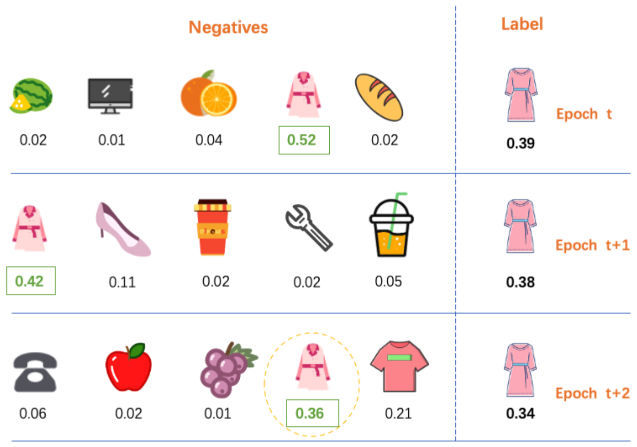

Figure 2: An illustrative example of how to mine a false negative sample from randomly selected negatives.

After that, we distinguish false negatives from set $\mathcal{N}_{rec}$, which consists of potential false negatives. Concretely, when negative samples are recorded over m epochs (a predefined hyper-parameter), we suppose these negative samples to be false negatives (i.e., set $\mathcal{N}_{false}$). A larger m means a stricter rule to identify false negatives and vice versa. Thus, if a negative sample obtains higher scores than the corresponding positive item over m times, we can conclude this sample is a potential positive sample. Figure 2 shows how to mine false negatives through their scores with $m = 3$.

False Negative Reversing. Recent researches [13, 23] commonly treat false negatives as noise and focus on reducing the risk of involving these samples. However, we believe false negatives can provide undiscovered user interests, and utilizing these samples can achieve better model performance than removing them. Specially, we treat false negatives in $\mathcal{N}_{false}$ as newly defined positive samples and reverse their original labels to train the model following Eq. 3.2 with $\boldsymbol{v}_{t+1}^{+} \in \mathcal{N}_{false}$.

3.3 False Negative Consistency Regularization (FCR)

To reduce model noises from reversed false negatives, we apply self-training to restrict false negatives in the output space. Concretely, we apply an EMA operation to generate a mean teacher from the current model and then use a consistency regularization loss to restrict predictions of false negatives.

Exponential Moving Average ($\boldsymbol{EMA}$). To improve label quality and avoid extra model noise from false negatives, we derive soft labels for these samples. Considering that an ensemble of historical models is more stable and accurate than a single model [19], we apply a teacher-student framework based on self-training to construct soft labels. Models updated with gradient descent over each batch are regarded as student models, and the teacher model applies the Exponential Moving Averaging ($\boldsymbol{EMA}$) weights of student models to emphasize the influence of newly updated student models. Concretely, the parameters ω'_t of the teacher model at time step t are updated from the corresponding student model parameters ω_t by $\boldsymbol{EMA}$:

$$\omega'_t = d\omega'_{t-1} + (1-d)\omega_t \tag{3.3}$$

where $d \in [0, 1]$ is a decay rate. Without changes in model structures, we introduce a teacher model that can aggregate representations from current student models and provide stable soft labels for false negatives.

Consistency Regularization Loss. To reduce extra noise from reversed false negatives, we utilize soft labels $\hat{y}$ from the stable teacher model to calibrate the outputs of the current model. For example, suppose a false negative brings unexpected noise when trained as a positive sample. In that case, its prediction of the current model will dramatically differ from that of the reliable teacher model. Thus, replacing y in Eq. (3.1) with $\hat{y}$, a consistency regularization loss to restrict the outputs of the current model is formalized as:

$$\widehat{l}_{u,t}(\boldsymbol{v_{t+1}};\omega) = -\hat{y}log(\sigma(\boldsymbol{s_{u,t}v_{t+1}})) - (1-\hat{y})log(1-\sigma(\boldsymbol{s_{u,t}v_{t+1}})) \tag{3.4}$$

$$\mathcal{L}_{con}(\boldsymbol{s_{u,t}};\omega) = \sum_{\boldsymbol{v_{t+1}^{+}} \in \mathcal{N}_{false}} \widehat{l}_{u,t}(\boldsymbol{v_{t+1}^{+}};\omega) \tag{3.5}$$

3.4 Final Objective.

Finally, we train the above consistency objective together with the backbone model's basic loss. The final objective is defined as:

$$\mathcal{L}_{final} = \mathcal{L}_{basic} + \alpha\mathcal{L}_{con} \tag{3.6}$$

where α is the coefficient weight to control the effect of $\mathcal{L}_{con}$. And the student model parameters are updated with the $\mathcal{L}_{final}$:

$$\omega = \omega - \gamma\frac{\partial\mathcal{L}_{final}}{\partial\omega} \tag{3.7}$$

,where γ is the learning rate.

3.5 Training Algorithm

Algorithm 1 shows the whole training process of UFNRec. As shown in lines 2-7, we only train with $\mathcal{L}_{basic}$ while the basic loss descends rapidly. Then, we start to generate teacher models with Eq. 3.3 (line 9). Here, negative samples are firstly collected from $\mathcal{N}_{rec}$ and then randomly sampled from the whole dataset (line 11). We filter out negative samples with larger scores than related positive samples and update $\mathcal{N}_{rec}$ (line 12). After that, we regard negative samples recorded over m times as false negatives and update $\mathcal{N}_{false}$ (line 13). Lines 14-16 calculate $\mathcal{L}_{con}$ only for $\boldsymbol{v}^{+}_{\boldsymbol{t+1}} \in \mathcal{N}_{false}$ and calculate $\mathcal{L}_{basic}$ for all positive samples including $\boldsymbol{v}^{+}_{\boldsymbol{t+1}} \in \mathcal{N}_{false}$. Lines 17-18 compute $\mathcal{L}_{final}$ based on Eq. 3.6 and update the model parameters. The whole training process will continue until convergence. For the computational complexity of UFNRec, the teacher model brings once feed-forward and parameter update. Besides, the number of false negative samples will influence the training time.

Algorithm 1 UFNRec Training Algorithm

```
Input: Training data $\mathcal{D} = \{s_{u_i,t}\}_{i=1}^{N}$
Output: model parameters $\omega$
 1: Initialization model with parameters $\omega$
 2: Initialization empty set $\mathcal{N}_{rec}$
 3: while $\mathcal{L}_{basic}$ descending rapidly do
 4:     $s_{u_i,t} \sim \mathcal{D}$
 5:     randomly sample $\boldsymbol{v_{t+1}}^{-} \in \mathcal{V}$
 6:     $g \leftarrow \bigtriangledown_{\omega}\mathcal{L}_{basic}$, $\omega \leftarrow GradientUpdate(\omega, g)$
 7: end while
 8: while not converged do
 9:     $s_{u_i,t} \sim \mathcal{D}$
10:     sample $\boldsymbol{v}^{-}_{\boldsymbol{t+1}} \in \mathcal{N}_{ran} \cup \mathcal{N}_{rec}$
11:     update $\mathcal{N}_{rec}$ based on prediction results $\sigma(\boldsymbol{s_{u,t}v_{t+1}})$
12:     update $\mathcal{N}_{false}$ based on parameter $m$
13:     update $\mathcal{L}_{basic}$ with Eq. 3.2 including $\boldsymbol{v}^{+}_{\boldsymbol{t+1}} \in \mathcal{N}_{false}$
14:     update teacher model parameters $\omega'$ with Eq. 3.3
15:     update $\mathcal{L}_{con}$ based on Eq. 3.5 for $\boldsymbol{v}^{+}_{\boldsymbol{t+1}} \in \mathcal{N}_{false}$
16:     $g \leftarrow \bigtriangledown_{\omega}\mathcal{L}_{final}$, $\omega \leftarrow GradientUpdate(\omega, g)$
17: end while
```

4 Experiments

4.1 Experiment Setup

4.1.1 Datasets Extensive experiments have been conducted on three widely used public benchmark datasets. We present their detailed statistics in Table 1.
Amazon. Datasets introduced by [15] product reviews collected from *Amazon.com.* We use top-level product categories to separate datasets and utilize the ***Beauty*** and ***Sports*** categories to evaluate model performance.
Yelp. is a well-known dataset for business recommendations, which is culled from the Yelp platform*. Following [17], we regard business categories as attributes and only leverage data after January 1st, 2019.

*https://www.yelp.com/dataset

Table 1: Dataset statistics of three public benchmarks, where *avg.* refers to the average actions per user.

Dataset	#users	#items	#actions	avg.	density
Beauty	52,024	57,289	0.4M	7.6	0.01%
Sports	25,598	18,357	0.3M	8.3	0.05%
Yelp	30,431	20,033	0.3M	10.4	0.05%

4.1.2 Baselines To prove the effectiveness of UFNRec, we choose three SOTA methods as baselines.

- **SASRec [1].** It utilizes the multi-head self-attention mechanism to tackle the SR task, and this model is widely regarded as one of the state-of-the-art baselines.
- **BERT4Rec [8].** It adopts a bidirectional self-attention mechanism to model user interaction sequences in the SR task. Like BERT [16], this model predicts the masked items in the historical sequences during training.
- **SSE-PT [18].** SSE-PT proposes a personalized transformer architecture with a novel regularization technique of stochastic shared embeddings.

4.1.3 Evaluation Metrics We choose the leave-one-out strategy to evaluate model performance, which has been widely used in many previous studies [1, 10]. Notably, each user's last interacted item will be used for testing. Similar to [1, 17], for each positive item, we randomly sample 100 items from the whole dataset and rank them by prediction scores. We evaluate model performance by HR@k and NDCG@k with $k = \{1, 5, 10\}$, which are commonly applied in SR tasks.

4.1.4 Hyper-Parameter Settings All models are implemented based on PyTorch with well-tested versions from the open-source community. We follow the original model settings of the correlated papers, i.e., the embedding dimension size. All models are optimized by Adam with a learning rate of 0.001, and the batch size is set to 128. For all datasets, the maximum sequence length is 50. We apply our UFNRec method to the above three backbone models by adding a false negative mining and reversing component (FMR) and a consistency regularization loss (FCR). For the false negative reversing step, we conduct experiments with m selected from $\{1, 2, 3, 4, 6, 8, 10\}$. To verify the influence of the loss weight α and the model decay rate d, we adjust the

Table 2: Model performance of backbones and our proposed UFNRec on three offline datasets, where '+UFN' refers to adding UFNRec to baselines, e.g., SASRec, BERT4Rec, SSE-PT. *Improv.* refers to the relative improvements of UFNRec over backbones, which are statistically significant with $p<0.05$.

Datesets	Metrics	SASrec	+UFN	Improv.	BERT4Rec	+UFN	Improv.	SSE-PT	+UFN	Improv.
Beauty	HR@1	0.2040	0.2175	6.62%	0.1919	0.1929	0.52%	0.2013	0.2256	12.07%
	HR@5	0.3785	0.3978	5.10%	0.3689	0.3860	4.64%	0.3884	0.4097	5.48%
	HR@10	0.4765	0.4953	3.95%	0.4690	0.4877	3.99%	0.4843	0.5002	3.28%
	NDCG@5	0.2942	0.3125	6.22%	0.2864	0.2919	1.92%	0.2990	0.3208	7.29%
	NDCG@10	0.3258	0.3439	5.56%	0.3187	0.3247	1.88%	0.3300	0.3500	6.06%
Sports	HR@1	0.1797	0.1909	6.23%	0.1619	0.1697	4.82%	0.1856	0.1951	5.12%
	HR@5	0.3952	0.4060	2.73%	0.3723	0.3841	3.17%	0.3982	0.4029	1.18%
	HR@10	0.5072	0.5120	0.95%	0.4936	0.5021	1.72%	0.5073	0.5126	1.04%
	NDCG@5	0.2925	0.3049	4.24%	0.2680	0.2775	3.54%	0.2970	0.3024	1.82%
	NDCG@10	0.3286	0.3392	3.23%	0.3071	0.3153	2.67%	0.3322	0.3378	1.69%
Yelp	HR@1	0.2743	0.2891	5.40%	0.2686	0.2954	9.98%	0.2979	0.3043	2.08%
	HR@5	0.6040	0.6239	3.29%	0.6090	0.6475	6.32%	0.6211	0.6351	2.22%
	HR@10	0.7589	0.7637	0.63%	0.7530	0.7857	4.34%	0.7554	0.7708	2.00%
	NDCG@5	0.4434	0.4681	5.57%	0.4527	0.4829	6.67%	0.4670	0.4773	2.15%
	NDCG@10	0.4934	0.5134	4.05%	0.4993	0.5276	5.67%	0.5106	0.5214	2.06%

values of α from $\{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and d from $\{0.9, 0.99, 0.995, 0.999\}$. Besides, we test the influence of different batch sizes from $\{32, 64, 128, 256, 512\}$.

4.2 Main Results As shown in Table 2, we report the experimental results of all models on three real-world datasets to compare their performance in sequential recommendation. Notice that UFNRec only introduces a false negative mining and reversing component (FMR) and a consistency training objective (FCR) into backbones without involving extra data or changing the network structure. We compare *UFNRec* with the three backbones to calibrate the influence of our method. We can observe that: 1) *SSE-PT* performs better than *SASRec* and *BERT4Rec* on most metrics, which indicates a personalized Transformer architecture is effective for sequential recommendation. Furthermore, *SSE-PT +UFN* can continuously improve the performance of *SSE-PT*, achieving 2.10% and 3.27% (on average) improvements in terms of HR@10 and NDCG@10, respectively. 2) UFNRec improves the performance over three baselines on all metrics. Compared with the randomly sampling strategy applied in backbones, our UFNRec can mine false negatives and utilize them to improve model performance. The universal enhancements of *UFNRec* over *SASRec*, *BERT4Rec*, and *SSE-PT* on three datasets with all HR@k and NDCG@k scores (relative improvements ranging from 0.63% to 12.07%) generally verify the effectiveness of UFNRec. Thus, UFNRec is complementary to these backbones and can further improve the performance of SOTA models.

Table 3: Ablation study of FMR and FCR on the Beauty dataset (p<0.05). '+FMR' refers to only adding the FMR operation to SASRec. '+FCR' refers to SASRec applying $\mathcal{L}_{final}$ without false negative reversing.

Metrics	SASRec	+FMR	+FCR	+UFN
HR@1	0.2040	0.2112	0.2081	0.2175
HR@5	0.3785	0.3903	0.3882	0.3978
HR@10	0.4765	0.4872	0.4836	0.4953
NDCG@5	0.2942	0.3044	0.3025	0.3125
NDCG@10	0.3258	0.3356	0.3333	0.3439

4.3 Ablation Study We conduct an ablation study to explore the influence of the FMR and FCR mentioned in Section 3. As shown in Table 3, the FMR operation contributes significant improvements over *SASRec* with a range from 1.87% to 3.53% on all metrics, which concludes that transferring false negatives to positive samples can indeed improve model performance. In contrast, overlooking false negatives can perturb the training process and yield inferior performance. Besides, to observe the effect of the consistency regularization objective, we maintain the false negative mining process and apply a consistency training objective to these samples without the label reversing step. Table 3 shows that FCR can also yield performance improvements (relative improvements ranging from 1.49% to 2.82% on all metrics), which indicates a consistency training objective based on self-training to regularize the output distributions of false negatives is essential. Furthermore, the combination of FMR and FCR can continuously en-

Table 4: Performance comparison of different strategies (i.e., removal vs. utilization) for false negative samples and different variants of UFNRec (i.e., +UFN$^{Rec}_{srns}$ and +UFN$_{srns}$) with varied false negative mining methods ($p<0.05$). Experiments are conducted on the Beauty dataset. Best performances are written in bold.

Strategies	origin	Removal			Utilization		
Metrics	SASRec	SASRec^R	+SRNS^{Rec}	+SRNS	+UFN$^{Rec}_{srns}$	+UFN$_{srns}$	+UFN
HR@1	0.2040	0.2119	0.2164	0.2062	0.2152	**0.2197**	0.2175
HR@5	0.3785	0.3886	0.3813	0.3879	0.3912	0.3864	**0.3978**
HR@10	0.4765	0.4821	0.4744	0.4886	0.4897	0.4831	**0.4953**
NDCG@5	0.2942	0.3024	0.3028	0.3004	0.3073	0.3071	**0.3125**
NDCG@10	0.3258	0.3333	0.3329	0.3328	0.3385	0.3381	**0.3439**

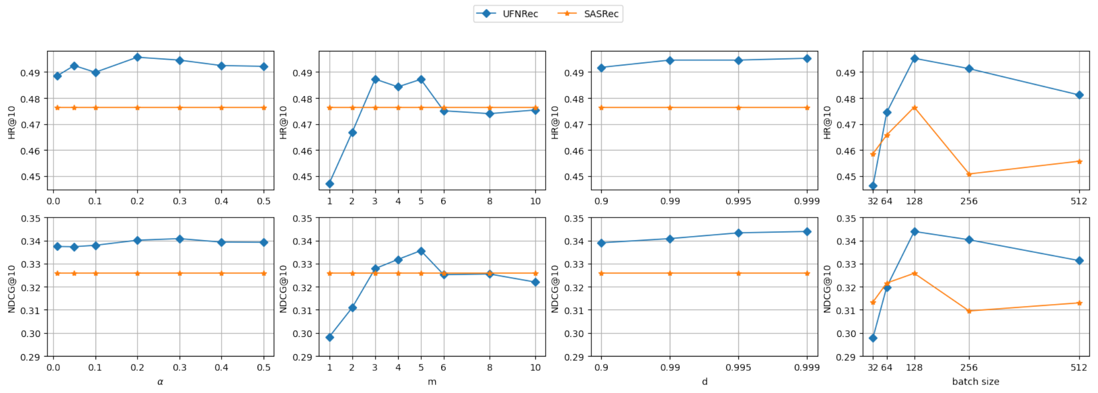

Figure 3: The impact of different hyper-parameters on HR@10 and NDCG@10 for UFNRec and SASRec on *Beauty*. Hyper-parameters include the loss weight α, the recorded times m, the decay rate d, and the batch size.

hance the model performance over a single component.

4.4 Analysis on False Negative Mining and Removing In this part, we compare UFNRec with the newly proposed SRNS [13], which introduces a variance-based negative sampling method but directly removes false negatives to reduce their influence. Here, we treat SRNS as an additional method to mine false negatives and compare the performance with UFNRec. As shown in Table 4, we conduct experiments on *Beauty* based on *SASRec* with three methods to remove false negatives and two variants of UFNRec to utilize them.

- **SASRec^R** utilizes the false negative mining method in Section 3.2 and removes false negatives.
- **+SRNS^{Rec}** applies the variance-based method proposed by SRNS [13] to distinguish false negatives from the set $\mathcal{N}_{rec}$ and removes these samples.
- **+SRNS** uses SRNS to mine false negatives without relying on $\mathcal{N}_{rec}$ and removes these samples.
- **+UFN$^{Rec}_{srns}$** applies SRNS to mine false negatives with $\mathcal{N}_{rec}$ and maintains all the remaining operations, including label reversing and the FCR part.
- **+UFN$_{srns}$** utilizes SRNS to mine false negatives without $\mathcal{N}_{rec}$ and maintains all the other parts.

As shown in Table 4, we compare the performance of different strategies for false negatives, including removal and utilization. Also, for UFNRec, we compare different variants of false negative mining methods. We can observe that: 1) The performance of *SASRec* is the worst on all metrics, which indicates overlooking false negatives will lead to inferior model performance. 2) Removal methods (e.g., *+SASRec^R*, *+SRNS^{Rec}*, and *+SRNS*) perform approximately better than *SASRec* by reducing the effect of involving false negatives. 3) Compared with simply removing these samples, *+UFN* outperforms all removal methods in all evaluation metrics by adding a label reversing step and a consistency training loss. 4) As for methods to utilize false negatives, *+UFN* performs better than *+UFN$^{Rec}_{srns}$* and *+UFN$_{srns}$* on most evaluation metrics, which proves the effectiveness of our false negative mining method.

4.5 Hyper-Parameter Analysis As shown in Figure 3, we mainly analyze the influence of four critical hyper-parameters for UFNRec. The first column in Figure 3 shows the consistency loss achieves significant improvements with a small α in Equation 3.6. As for the

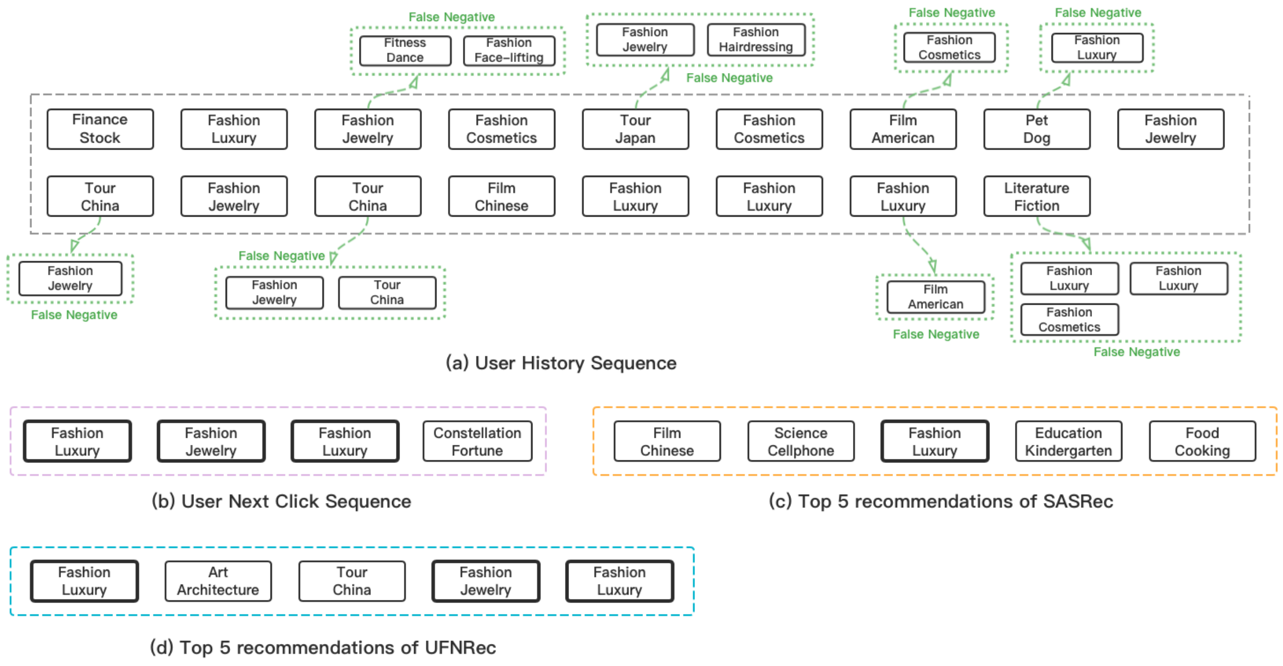

Figure 4: Case study on sequential recommendation. For simplicity, items are presented by their categories. (a) A historical sequence of a randomly sampled user from a real-world platform. Items in green boxes are false negatives mined by UFNRec. (b) The user's next click sequence after the historical sequence. (c) and (d) are the top 5 recommendation items of SASRec and UFNRec, respectively. Items belonging to Fashion are in bold boxes.

Table 5: Online A/B Tests ($p<0.05$).

Scenario	CVR	VPM
Tab Page	+7.94%	+5.83%
First Page	+8.60%	+7.00%

reversing parameter m, HR@10 and NDCG@10 first increase and then decrease with the increase of m, since a small m indicates a low criterion to identify false negatives and vice versa. The third column shows the model performance is approximately stable with a decay rate d in Equation 3.3 from 0.9 to 0.999. Also, though SASRec performs best at batch size=128, UFNRec can continuously improve the performance of SASRec.

4.6 Online A/B Test

We deploy UFNRec on a real-world recommendation platform used by over 300 million users to evaluate online performance.

Online Serving and Evaluation Protocol. The online matching component consists of rule-based approaches and embedding-based approaches. Concretely, we utilize UFNRec as an additional embedding-based retrieval approach, with other matching approaches unchanged. Since the online data is sufficient and the online model is stable enough, we directly set $m = 1$ here. In the online experiment, we deploy UFNRec on two recommendation scenarios with two metrics, including CVR and value per thousand impressions (VPM). We conduct the online A/B test for 5 days with 330 thousand users in the *Tab Page* scenario and almost one million users in the *First Page* scenario.

Experimental Results. Table 5 shows the relative improvements of UFNRec, from which we can observe that: 1) UFNRec achieves remarkable improvements on CVR and VPM metrics in both scenarios, which proves the effectiveness of UFNRec in online serving. 2) The *Tab Page* is a small scenario with few active users and relatively sparse user interactions, while UFNRec can extend user interest from false negatives and enhance the recommendation performance. 3) The improvement of VPM means that users attend to pay more in our system since UFNRec can draw user interests more accurately. The increment of CVR indicates that our system is attractive to more users.

4.7 Case Study

A crucial novelty of our model is that we realize the importance of false negatives and utilize them to improve the sequential recommendation performance. To better understand how UFNRec works, we qualitatively analyze a case from a real-world platform. As shown at the top of Figure 4, we randomly select one user and her historical click sequence, which contains many items belonging to the Fashion category and shows the user's interest in Fashion. As for 100 negative samples randomly selected for each clicked item, false negatives mined by UFNRec are mainly fashion items. Thus, training the model with these negative samples will disturb the training process. As shown in Figure 4(c) and Figure 4(d), UFNRec recommends three fashion items, while SASRec only recommends one. Compared with the user's next click sequence in Figure 4(b), UFNRec recommends more reasonably than SASRec. Obviously, UFNRec can discover false negatives hidden in negative samples and further utilize them to learn user interests. In contrast, SASRec overlooks the existence of false negatives and roughly trains the model with all randomly selected negative samples, which leads to a less robust model. Therefore, by utilizing false negatives, UFNRec can capture user interests more accurately and improve model performance.

5 Conclusion

In this paper, we propose a simple yet effective method UFNRec to utilize false negatives for SR tasks, which involves a false negative reversing step and a consistency training loss. We first design a strategy to distinguish false negatives from true negatives and transfer false negatives to positive samples to train the model further. Then, we introduce a teacher-student framework to provide soft labels for such false negatives and then apply a consistency loss to regularize outputs from the framework. Extensive experiments demonstrate the effectiveness and compatibility of our model. To the best of our knowledge, this is the first work to utilize false negatives to improve model performance for SR. In the near future, we will explore more false negative mining methods for SR.

References

[1] W.-C. Kang and J. McAuley, Self-attentive sequential recommendation, in *ICDM*, 2018, pp. 197–206.

[2] J.-T. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang, Embedding-based retrieval in facebook search, in *KDD*, 2020, pp. 2553–2561.

[3] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in *WWW*, 2010, pp. 811–820.

[4] R. He and J. McAuley, Fusing similarity models with markov chains for sparse sequential recommendation, in *ICDM*, 2016, pp. 191–200.

[5] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, Session-based recommendations with recurrent neural networks,*arXiv preprint arXiv:1511.06939*, 2015.

[6] J. Tang and K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in *WSDM*, 2018, pp. 565–573.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, *NIPS*, 2017, pp. 5998–6008.

[8] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in *CIKM*, 2019, pp. 1441–1450.

[9] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, Session-based recommendation with graph neural networks, in *AAAI*, 2019, pp. 346–353.

[10] C. Liu, X. Liu, R. Zheng, L. Zhang, X. Liang, J. Li, L. Wu, M. Zhang, and L. Lin, C^2-rec: An effective consistency constraint for sequential recommendation, *arXiv preprint arXiv:2112.06668*, 2021.

[11] P. Covington, J. Adams, and E. Sargin, Deep neural networks for youtube recommendations, in *RECSYS*, 2016, pp. 191–198.

[12] J. Yang, X. Yi, D. Zhiyuan Cheng, L. Hong, Y. Li, S. Xiaoming Wang, T. Xu, and E. H. Chi, Mixed negative sampling for learning two-tower neural networks in recommendations, in *WWW*, 2020, pp. 441–447.

[13] J. Ding, Y. Quan, Q. Yao, Y. Li, and D. Jin, Simplify and robustify negative sampling for implicit collaborative filtering, *NIPS*, vol. 33, pp. 1094–1105, 2020.

[14] J. Wang, J. Zhu, and X. He, Cross-batch negative sampling for training two-tower recommenders, in *SIGIR*, 2021, pp. 1632–1636.

[15] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, Image-based recommendations on styles and substitutes, in *SIGIR*, 2015, pp. 43–52.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in *NAACL*, 2019, pp. 4171–4186.

[17] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R.Wen, S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization, in *CIKM*, 2020, pp. 1893–1902.

[18] L. Wu, S. Li, C.-J. Hsieh, and J. Sharpnack, Sse-pt: Sequential recommendation via personalized transformer, in *RECSYS*, 2020, pp. 328–337.

[19] A. Tarvainen and H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, *NIPS*, vol. 30, 2017.

[20] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, Modeling user exposure in recommendation, in *WWW*, 2016, pp. 951–961.

[21] Q. Zhao, M. C. Willemsen, G. Adomavicius, F. M. Harper, and J. A. Konstan, Interpreting user inaction in recommender systems, in *RECSYS*, 2018, pp. 40–48.

[22] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, Probabilistic matrix factorization with non-random missing data, in *ICML*, 2014, pp. 1512–1520.

[23] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: simple and efficient negative sampling for knowledge graph embedding," in *ICDE*, 2019, pp. 614–625.