SEQEDITOR: EDITING USER BEHAVIOR SEQUENCE FOR RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

User Behavior Sequences (UBSs) are the data foundation for Sequential Recommender Systems (SRSs). Existing studies typically regard these sequences as high-quality behavioral features. However, UBSs in real-world scenarios often suffer from the *quality flaw problem*, manifested as noisy and missing items. This problem prevents SRSs from being fully optimized, hindering model performance. To address this issue, we propose **SeqEditor** to improve the quality of UBSs by editing sequences. Concretely, SeqEditor comprises a *Quality Assessor* and a *Quality Editor*. The Quality Assessor evaluates UBS quality by estimating local training utility from the alignment between sequence gradients and the prediction-loss gradient on held-out reference users. The LLM-based Quality Editor is optimized by performing supervised fine-tuning (SFT) and reinforcement learning (RL) with interest-aware rewards. This enables it to understand user behavior patterns and identify noise. During the inference stage, SeqEditor effectively improves UBS quality by performing potential missing item supplement and noise removal. SeqEditor can be applied to various SRSs, while its text-based Quality Editor supports cross-domain generalization. Extensive empirical results demonstrate the effectiveness of SeqEditor. Our anonymous code is available at https://anonymous.4open.science/r/SeqEditor_2027.

1 INTRODUCTION

In online service platforms, user behavior sequences (UBSs) generated from user actions like clicks or purchases form the data foundation for sequential recommender systems (SRSs). Researchers leverage UBSs as rich behavioral features and have developed numerous advanced SRSs. For instance, discriminative-based methods, e.g., SASRec (Kang & McAuley, 2018) and BERT4Rec (Sun et al., 2019), use UBSs as discriminative evidence, optimizing the models by maximizing the distance between positive and negative samples. In contrast, generative-based efforts like TIGER (Rajput et al., 2023) and HSTU (Zhai et al., 2024) regard UBSs as generative trajectories and model the joint distribution of sequences in an autoregressive manner. Despite following different technical routes, these works share a common assumption—every UBS is high-quality.

*However, UBSs in real-world scenarios often suffer from the **quality flaw problem***. Specifically, distinct from natural language sequences governed by strict grammatical rules, UBSs stem from diverse circumstances rather than universal patterns. This may lead to *noisy and missing* items in UBSs. The noisy items typically arise from users' error or accidental clicks. For the missing items, they are often caused by exposure bias (Abdollahpouri & Mansoury, 2020) or users' offline purchases. This leads to incomplete logical chains and semantic loss of user preference. The quality flaw problem hinders the adequate optimization of SRSs, limiting their performance upper bound.

To examine the impact of the quality flaw problem on SRSs, we perform the following empirical studies. Concretely, we first divide UBSs from the real-world dataset Beauty into high- and low-quality subsets of equal size (detailed in Sec. 3.1 and Appendix B.1). The latter usually contain noisy and missing items. We then analyze the attention-based SRS (SASRec) from both optimization and performance perspectives.

- For the optimization perspective, we optimize two separate SASRec on high- and low-quality UBSs, resulting in SASRec-HQ and SASRec-LQ, respectively. We then visualize their loss landscapes (Li et al., 2018) to compare the geometric properties of their convergence regions. As shown in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Figure 1-(a), SASRec-LQ exhibits a rougher loss landscape and converges to a suboptimal point, which demonstrates that low-quality UBSs hinder the model's effective optimization.

▷ From the performance perspective, we apply the well-trained SASRec-HQ and SASRec-LQ to evaluate the recommendation performance on the same test set, respectively. Figure 1-(b) shows that SASRec-LQ performs significantly worse than SASRec-HQ.

Although some studies (Kang & McAuley, 2018; Han et al., 2024; Zhang et al., 2024b) employ attention mechanisms or contrastive learning to mitigate the impact of noisy items, they still face two challenges: (i) They overlook the assessment of UBS quality, thus failing to effectively distinguish and leverage high-quality UBSs. (ii) Attention mechanisms or contrastive learning only attenuate rather than entirely eliminate noise, and fail to address the missing item issue.

To tackle these challenges, we propose **SeqEditor**, a model-agnostic framework, to improve the quality of UBSs. Specifically, SeqEditor comprises two core modules, including ① *Quality Assessor* and ② *Quality Editor*. The Quality Assessor uses a lightweight auxiliary recommender to score each UBS based on the alignment between its loss gradient and the prediction-loss gradient on held-out reference users. A high-quality subset of UBSs is identified based on this score and then labeled by introducing noise. Based on the labeled data, the LLM-based Quality Editor first undergoes supervised fine-tuning (SFT), followed by reinforcement learning (RL) with interest-aware rewards. This process equips Quality Editor with a deep understanding of user behavior patterns, allowing for the elimination of noisy items. With the optimized Quality Editor, we perform sequence editing during inference. This process supplements potential missing items based on directed mutual information between items and eliminates noisy items, thereby improving UBS quality. SeqEditor can be applied to various SRSs, while its text-based Quality Editor supports cross-domain generalization.

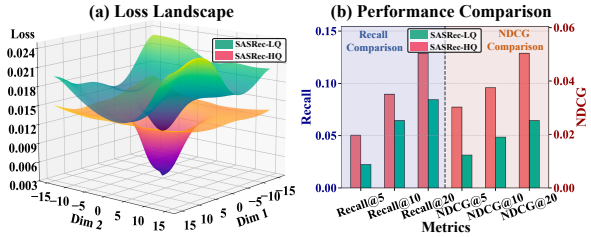

Figure 1: The loss landscapes and performance of SASRec-HQ and SASRec-LQ on the Beauty dataset.

- We propose SeqEditor, a model-agnostic framework that improves UBS quality by supplementing potentially missing items and eliminating noisy items. This framework enhances various SRSs and enables cross-domain generalization.
- We propose a gradient-based Quality Assessor that evaluates UBS quality through local training utility and selects high-utility seeds for training the Quality Editor.
- We propose the Quality Editor to understand user behavior patterns and eliminate noisy items through fine-tuning LLM with SFT and RL.

Extensive experiments on three benchmark datasets demonstrate that SeqEditor consistently enhances five SRSs, achieving an average improvement of 10.55% across them.

2 PRELIMINARY

In this section, we first delineate the fundamental concepts and symbols used throughout this paper, and then formalize the research problem.

Concepts and Symbols. Formally, a UBS, denoted as $S = [v_1, v_2, \cdots, v_n]$, describes the user's historical interactions with n items in chronological order. Here, $v_i \in \mathcal{V}$ represents the i-th item that the user interacts with, and $\mathcal{V}$ denotes the item set. In addition, for a given item v_i, it is associated with a text description and category information, denoted as $\mathcal{T}_i = [t_1, t_2, \cdots, t_m]$ and c_i, respectively. t_j signifies the j-th token in $\mathcal{T}_i$ (including m tokens). For additional notations, we will introduce them in the corresponding section.

Problem Formalization. The formal objective of this work is to optimize a sequence editing function $\mathcal{F}_\theta(\cdot) : S \to \widetilde{S}$. It aims to produce an edited sequence $\widetilde{S}$ with a higher quality score $Q(\widetilde{S})$, thereby improving the performance $\mathcal{P}(\mathcal{M}_\phi(\cdot))$ of recommendation model $\mathcal{M}_\phi(\cdot)$. This goal can be

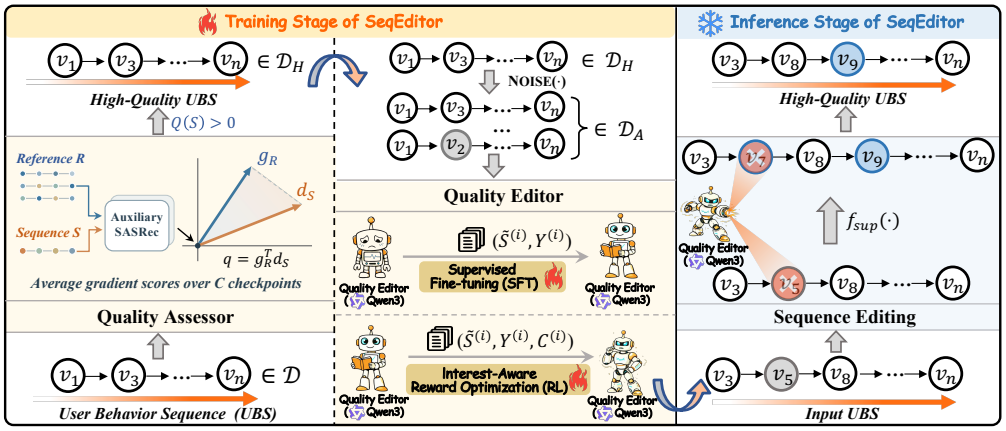

Figure 2: The framework of SeqEditor, which consists of two core components, including Quality Assessor and Quality Editor. The inference pipeline of SeqEditor is called Sequence Editing.

formulated as a constrained optimization problem:

$$\max_{\theta} \ \mathbb{E}_{S\sim\mathcal{D}}\Big[Q\big(\mathcal{F}_\theta(S)\big)\Big]$$
$$s.t.\ \mathbb{E}_{S\sim\mathcal{D}}\Big[\mathcal{P}\big(\mathcal{M}_\phi(\mathcal{F}_\theta(S))\big)\Big] > \mathbb{E}_{S\sim\mathcal{D}}\Big[\mathcal{P}\big(\mathcal{M}_\phi(S)\big)\Big], \tag{1}$$

where $\mathcal{D}$ is the training set.

3 Methodology

In this section, we present SeqEditor, a model-agnostic framework designed to improve the quality of UBSs. SeqEditor consists of two components, including (i) *Quality Assessor* and (ii) *Quality Editor*. Its overall architecture is demonstrated in Figure 2.

3.1 Quality Assessor

The Quality Assessor aims to evaluate the quality of UBSs, thereby identifying a high-quality subset for optimizing SeqEditor. Guided by the recommendation objective, we define sequence quality as its local contribution to improving predictions for held-out reference users. Specifically, a sequence has positive training utility if a small training step on it reduces the model's prediction loss on reference users. We estimate sequence quality using gradients computed with a lightweight auxiliary recommender.

Reference and Auxiliary Training Sets. We first group users by the length of their training UBSs and randomly sample the same proportion from each group to form the initial reference set $\mathcal{R}$. The remaining users' UBSs form the auxiliary training set $\mathcal{D}$. To prevent items unseen during auxiliary training from affecting the reference loss, we check item coverage and require $I(\mathcal{R}) \subseteq I(\mathcal{D})$, where $I(\cdot)$ denotes the set of items in the corresponding dataset.

Auxiliary Recommender and Prediction Objective. We use SASRec as the auxiliary recommender $p_\theta(\cdot)$. We train the model on $\mathcal{D}$ and save checkpoints at predefined epochs for subsequent quality scoring. All model parameters are optimized during training. During scoring, we keep the model parameters fixed and compute gradients only with respect to the shared sequence encoder parameters.

For a UBS $S = [v_1, \ldots, v_n] \in \mathcal{D}$, we define the sequence loss $\ell(S;\theta)$ as the average cross-entropy over all prediction positions. The reference objective $\mathcal{L}_\mathcal{R}(\theta)$ is the average prediction loss across all users in the reference set. They are defined as follows:

$$\mathcal{L}_\mathcal{R}(\theta) = \frac{1}{|\mathcal{R}|}\sum_{S_u\in\mathcal{R}} \ell(S_u;\theta), \quad \ell(S;\theta) = -\frac{1}{n-1}\sum_{t=1}^{n-1} \log p_\theta(v_{t+1} \mid v_1, \ldots, v_t). \tag{2}$$

where $S_u \in \mathcal{R}$. $\mathcal{L}_{\mathcal{R}}(\theta)$ reflects the model's average prediction performance on the reference users.

Gradient-based Quality Scoring. For a sequence $S \in \mathcal{D}$, we estimate how further training on it would affect the prediction loss on reference users. Specifically, at the same auxiliary model checkpoint, we compute the gradients of the sequence loss and the reference loss:

$$g_S = \nabla_\theta \ell(S;\theta), \qquad g_{\mathcal{R}} = \nabla_\theta \mathcal{L}_{\mathcal{R}}(\theta). \tag{3}$$

Here, $-g_S$ is the direction that reduces the prediction loss of sequence S, while $-g_{\mathcal{R}}$ reduces the reference loss. **When the directions of g_S and $g_{\mathcal{R}}$ are positively aligned, a small update along $-g_S$ can also improve predictions for reference users.** To control for differences in gradient magnitude across sequences, we further normalize the sequence gradient and compute its inner product with the reference gradient:

$$q(S;\theta) = g_{\mathcal{R}}^\top d_S, \qquad d_S = \frac{g_S}{\|g_S\|_2 + \epsilon}, \tag{4}$$

where $\epsilon > 0$ is a numerical stability term. The score $q(S;\theta)$ measures the first-order decrease in the reference loss $\mathcal{L}_{\mathcal{R}}(\theta)$ per unit step along the normalized descent direction $-d_S$.

Specifically, consider a virtual update $\theta' = \theta - \alpha d_S$ with step size $\alpha > 0$. If the gradient of the reference loss is locally Lipschitz continuous, then

$$\mathcal{L}_{\mathcal{R}}(\theta') - \mathcal{L}_{\mathcal{R}}(\theta) = -\alpha q(S;\theta) + O(\alpha^2). \tag{5}$$

Thus, when $q(S;\theta) > 0$ and the step size is sufficiently small, this update reduces the reference prediction loss. For the same model state and step size, a larger $q(S;\theta)$ indicates a greater first-order loss reduction. This directly links sequence quality to its local training utility for the reference prediction objective.

The training utility of a sequence $S \in \mathcal{D}$ may vary across training stages. We therefore save model parameters at the end of C predefined epochs, compute a score at each checkpoint, and average the scores. Therefore, we define the quality score of sequence $S \in \mathcal{D}$ as $Q(S)$:

$$Q(S) = \frac{1}{C} \sum_{c=1}^{C} q(S;\theta^{(c)}), \tag{6}$$

where C is the number of checkpoints used for scoring, and $\theta^{(c)}$ denotes the shared encoder parameters at the k-th saved checkpoint. All checkpoints come from a single training run of the auxiliary recommender.

After computing $Q(S)$ for each UBS $S \in \mathcal{D}$, we select UBSs with $Q(S) > 0$ as high-quality set $\mathcal{D}_H$.

3.2 Quality Editor

The Quality Assessor relies on item ID representations learned by the auxiliary recommender, limiting the applicability of its scores to the training item set. To enable generalizable sequence editing, we use the Quality Assessor to select high-quality set $\mathcal{D}_H$ and introduce an LLM-based Quality Editor. Given item text and sequence context, the Editor learns an editing policy to improve sequence quality without relying on fixed item ID embeddings. Specifically, Quality Editor comprises two steps: (i) Annotated Data Construction and (ii) Quality Editor Optimization.

3.2.1 Annotated Data Construction

For a UBS $S = [v_1, v_2, \cdots, v_n] \in \mathcal{D}_H$, we randomly add 0 to K ($K = 5$ in our setting) noise items that do not exist in S, thus constructing a perturbed UBS $\bar{S}$ with the corresponding label Y (the introduced noise items). Thus, the annotated training set $\mathcal{D}_A$ is constructed as:

$$\mathcal{D}_A = \{(\bar{S}, Y) \mid \bar{S}, Y = \textsc{Noise}(S), S \in \mathcal{D}_H\}, \tag{7}$$

where $\textsc{Noise}(\cdot)$ denotes the operation that inserts random noise items.

We adopt LLM as the foundation model for the Quality Editor and construct its training samples based on $\mathcal{D}_A$. Specifically, for each perturbed sequence $\bar{S}$, we integrate the text descriptions and category information of its items into a unified prompt. The corresponding Y is used as the training label. By employing generic natural language as input, SeqEditor gains the capability of cross-domain generalization. Figure 7 in Appendix B.2 provides a schematic illustration of the training samples.

3.2.2 QUALITY EDITOR OPTIMIZATION

Based on the above training samples, we perform supervised fine-tuning (SFT) of LLM using LoRA (Hu et al., 2022). This process enables Quality Editor to understand user behavior patterns and identify noise items. In addition, SFT improves the instruction-following capability of LLMs in this task. To ensure inference efficiency given the large and ever-growing volume of daily UBSs, we select the compact Qwen3-4B (Yang et al., 2025a) as our base LLM. We optimize the negative log-likelihood loss $\mathcal{L}_{\text{SFT}}$ for SFT:

$$\mathcal{L}_{SFT} = \mathbb{E}_{(\tilde{S},Y)\sim\mathcal{D}_A}\Big[-\sum_{t=1}^{|Y_{tokens}|}\log P_\theta(y_t|\mathcal{X}(\tilde{S}), y_{1:t-1})\Big], \tag{8}$$

where $\mathcal{X}(\cdot)$ is the prompt function, Y_{tokens} is the text token sequence corresponding to the label Y and $y_t \in Y_{tokens}$ is the t-th token. P_θ signifies the conditional probability of LLM under low-rank adaptation parameters θ.

Despite the performance gains from SFT, the resulting model still suffers from two shortcomings: (i) On difficult samples, the model still requires multi-times inference to arrive at the correct answer, resulting in unstable performance and high computational costs; (ii) Different users have varying scopes of interest. Broad interests often correspond to more complex behavioral patterns that blur the line between normal behavior and noise. Such UBSs may cause normal items to be misclassified as noise, leading to an increase in the number of false positives. To address these issues, we propose an RL-based optimization called *Interest-Aware Reward Optimization*.

Interest-Aware Reward Optimization. RL optimizes decision-making through trial and feedback, improving the model's performance of single-pass inference (Yue et al., 2025). Furthermore, we design an interest-aware reward function $R^{int}(\cdot)$ to steer SeqEditor toward greater sensitivity to users with diverse interests, thereby reducing false positive predictions in such samples.

Given a UBS $S = [v_1, v_2, \cdots, v_n]$, each item v_j corresponds to a category c_j. The categories of all items within S, denoted as $C = \{c_1, c_2, \cdots, c_k\}$, reflect the range of user interests—a larger $|C|$ corresponds to broader interests. The interest-aware reward function $R^{int}(\cdot)$ extends the F1 score by incorporating a penalty term $-\frac{(|C|-1)}{(|C|+1)}\frac{FP}{(TP+FP)}$ that grows with interest breadth, defined as:

$$R^{int}(\hat{Y}, Y, C) = F_1 - \frac{(|C|-1)}{(|C|+1)}\frac{FP}{(TP+FP)}, \tag{9}$$

where Y and $\hat{Y}$ are the true and predicted noise items; TP (true positives) and FP (false positives) correspond to correct and incorrect noise predictions. The penalty term increases with $|C|$, imposing a stricter cost for false positives when user interests are broader.

Thereafter, we employ the RL algorithm DAPO (Yu et al., 2025) to perform the optimization, its loss function $\mathcal{L}_{DAPO}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{DAPO} =& \mathbb{E}_{(\tilde{S},Y)\sim\mathcal{D}_\mathcal{A},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(\cdot|\mathcal{X}(\tilde{S}))}\Bigg[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\big(r_{i,t}(\theta),\\ & 1-\varepsilon_{low}, 1+\varepsilon_{high}\big)\hat{A}_{i,t}\Big)\Bigg], \quad s.t.\ \ 0 < |\{o_i \mid \text{is_equivalent}(Y_{tokens}, o_i)\}| < G.\end{aligned} \tag{10}$$

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid \tilde{S}, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t} \mid \tilde{S}, o_{i,<t})}, \hat{A}_{i,t} = \frac{R_i^{int} - mean(\{R_i^{int}\}_{i=1}^{G})}{std(\{R^{int}\}_{i=1}^{G})}. \tag{11}$$

3.3 SEQUENCE EDITING

This section presents the inference stage of SeqEditor for improving the quality of UBSs. Specifically, given an input UBS S (which may be low-quality or newly formed), we first apply the Quality Editor, denoted as $\text{QE}(\cdot)$, to identify noise items, which are then removed via a denoising function $f_{den}(\cdot)$, yielding the denoised UBS S_{den}:

$$S_{den} = f_{den}\big(S, \text{QE}(\mathcal{X}(\mathcal{S}))\big), \tag{12}$$

where $f_{den}(\cdot)$ operates on S by filtering out items identified as noise and $\mathcal{X}(\cdot)$ is the prompt function.

After denoising, we address missing items by supplementing S_{den}. To identify potential predecessors for insertion, we compute a direction-aware pointwise mutual information (PMI) score from ordered adjacent item pairs. All statistics are collected from the original training sequences, excluding validation and test interactions:

$$s^{\text{PMI}}_{v_i \to v_j} = \sigma\Big(\log \frac{M\, C(v_i \to v_j)}{C_{\text{out}}(v_i)\, C_{\text{in}}(v_j)} \Big), \tag{13}$$

where $C(v_i \to v_j)$ counts the occurrences of v_i immediately followed by v_j, and M is the total number of adjacent item pairs. σ is the Sigmoid function. The marginal counts are defined as

$$C_{\text{out}}(v_i) = \sum_{v} C(v_i \to v), \qquad C_{\text{in}}(v_j) = \sum_{v} C(v \to v_j). \tag{14}$$

A higher score $s^{\text{PMI}}_{v_i \to v_j} \in (0, 1)$ indicates a stronger statistical association for v_i immediately preceding v_j. Unobserved transitions are assigned a score of 0.

Thereafter, for each item $v_j \in S_{den}$, a supplementation function $f_{sup}(\cdot)$ uses $s^{\text{PMI}}_{v_i \to v_j}$ to insert potential missing items, producing an augmented UBS S_{sup}. Since this supplementation is based on statistical correlations, S_{sup} may contain newly introduced noise. Therefore, we apply the Quality Editor and the denoising function $f_{den}(\cdot)$ again (as in Eq. (12)) to filter them out, yielding the final high-quality UBS $\widetilde{S}$. The entire pipeline is formalized as:

$$\widetilde{S} = f_{\text{den}}\big(S_{\text{sup}}, \text{QE}(\mathcal{X}(S_{\text{sup}}))\big), \tag{15}$$

$$S_{\text{sup}} = \big[f_{\text{sup}}(v_1, \mathcal{V}_{\text{tr}}), \ldots, f_{\text{sup}}(v_m, \mathcal{V}_{\text{tr}})\big], \tag{16}$$

$$v_k^* = \underset{v_k \in \mathcal{V}_{\text{tr}} \setminus S}{\arg\max}\, s^{\text{PMI}}_{v_k \to v_i}, \quad f_{\text{sup}}(v_i, \mathcal{V}_{\text{tr}}) = \begin{cases} (v_k^*, v_i), & \text{if } s^{\text{PMI}}_{v_k^* \to v_i} > \eta, \\ (v_i), & \text{otherwise.} \end{cases} \tag{17}$$

Here, $\mathcal{V}_{\text{tr}}$ denotes the items observed in the training interactions, and η is a hyperparameter.

Notably, the sequence editing pipeline is performed as "denoising→supplementing→denoising", rather than "supplementing→denoising". The rationale is that performing supplementation first may expand the noise present in UBSs, forming complete interest blocks that are difficult for the SeqEditor to identify.

Based on the high-quality UBS $\widetilde{S}$, we optimize SRSs by minimizing loss such as binary cross-entropy to perform recommendation.

4 Experiments

4.1 Experimental Settings

Datasets. We adopt three datasets from the Amazon Reviews (McAuley et al., 2015) for our experiments, including "Clothing, Shoes and Jewelry" (*Clothing*), "Sports and Outdoors" (*Sports*), and "Beauty" (*Beauty*). Detailed introductions are provided in Appendix C.1.

Evaluation Metrics. The performance of all SRSs is evaluated using two standard metrics: Recall@K and NDCG@K, where $K \in \{5, 10\}$.

Recommendation Models. We equip SeqEditor with two types of SRSs, including discriminative recommendation (DR)-based models and generative recommendation (GR)-based models, to examine their performance variations. Specifically, DR-based models comprise GRU4Rec (Hidasi et al., 2015), SASRec (Kang & McAuley, 2018) and BERT4Rec (Sun et al., 2019), while GR-based models include TIGER (Rajput et al., 2023) and HSTU (Zhai et al., 2024). Their detailed descriptions are provided in Appendix C.2.

4.2 Overall Performance

To evaluate the performance of SeqEditor, we compare the overall performance of five SRSs and their corresponding versions equipped with SeqEditor. Based on these results shown in Table 1 , the following conclusions can be drawn.

Table 1: The overall performance of five baselines and their enhanced versions with SeqEditor on three benchmark datasets. Within each block (corresponding to a baseline), the best-performing result is shown in bold, and its relative improvement over the baseline is highlighted in a blue background. All reported results are averaged over 5 independent runs. A mark † denotes an improvement that is statistically significant over the baseline (p-value< 0.05, t-test).

Methods	Beauty				Sports				Clothing			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
GRU4Rec	0.0164	0.0283	0.0099	0.0137	0.0129	0.0204	0.0086	0.0110	0.0157	0.0248	0.0088	0.0117
SeqEditor+GRU4Rec	**0.0208**†	**0.0373**†	**0.0123**†	**0.0177**†	**0.0168**†	**0.0245**†	**0.0097**†	**0.0118**†	**0.0173**†	**0.0258**†	**0.0097**†	**0.0126**†
Imp(%)	26.83%	31.80%	24.24%	29.20%	30.23%	20.10%	12.80%	7.27%	10.19%	4.03%	10.23%	7.69%
BERT4Rec	0.0203	0.0347	0.0124	0.0170	0.0115	0.0191	0.0075	0.0099	0.0140	0.0225	0.0076	0.0102
SeqEditor+BERT4Rec	**0.0264**†	**0.0409**†	**0.0128**†	**0.0179**†	**0.0140**†	**0.0215**†	**0.0083**†	**0.0104**†	**0.0154**†	**0.0240**†	**0.0084**†	**0.0111**†
Imp(%)	30.05%	17.88%	3.23%	5.29%	21.74%	12.56%	10.67%	5.05%	10.00%	6.67%	10.53%	8.82%
SASRec	0.0387	0.0605	0.0249	0.0318	0.0233	0.0350	0.0154	0.0192	0.0187	0.0272	0.0103	0.0130
SeqEditor+SASRec	**0.0430**†	**0.0658**†	**0.0260**†	**0.0341**†	**0.0266**†	**0.0414**†	**0.0171**†	**0.0214**†	**0.0205**†	**0.0308**†	**0.0112**†	**0.0142**†
Imp(%)	11.11%	8.76%	4.42%	7.23%	14.16%	18.29%	11.04%	11.46%	9.63%	13.23%	8.74%	9.23%
TIGER	0.0454	0.0648	0.0321	0.0384	0.0264	0.0400	0.0181	0.0225	0.0192	0.0301	0.0121	0.0155
SeqEditor+TIGER	**0.0487**†	**0.0682**†	**0.0336**†	**0.0404**†	**0.0282**†	**0.0443**†	**0.0192**†	**0.0239**†	**0.0201**†	**0.0320**†	**0.0133**†	**0.0169**†
Imp(%)	7.27%	5.25%	4.67%	5.21%	6.82%	10.75%	6.08%	6.22%	4.69%	6.31%	9.92%	9.03%
HSTU	0.0469	0.0704	0.0314	0.0389	0.0258	0.0414	0.0165	0.0215	0.0211	0.0320	0.0111	0.0146
SeqEditor+HSTU	**0.0490**†	**0.0712**†	**0.0334**†	**0.0410**†	**0.0274**†	**0.0439**†	**0.0181**†	**0.0228**†	**0.0223**†	**0.0326**†	**0.0120**†	**0.0154**†
Imp(%)	4.48%	1.14%	6.37%	5.40%	6.20%	6.04%	9.70%	6.05%	5.69%	1.88%	8.11%	5.48%

Benefiting from the effectiveness and model-agnostic design of SeqEditor, it significantly improves the Recall and NDCG of all five SRSs. Specifically, SeqEditor brings average performance gains of 11.99%, 11.66%, and 8.00% to five SRSs across the Beauty, Sports, and Clothing datasets, respectively. These improvements stem from SeqEditor's ability to boost UBS quality by alleviating the quality flaw problem, providing a better data foundation for the optimization of various SRSs.

Further analysis of SeqEditor's performance across different datasets, we observe the greatest improvement on the Beauty dataset. This can be attributed to Beauty containing a smaller number of items and categories; correspondingly, its UBSs exhibit a narrower scope of interests. These reduce the difficulty of noise identification and information supplementation.

In addition, the performance comparison between SeqEditor with recent denoising-based models (Zhou et al., 2023; Zhang et al., 2024a; Wang et al., 2025) are shown in Appendix C.8. The experimental results demonstrate the clear superiority of SeqEditor.

Table 2: Ablation Study on SeqEditor

Variant Models	Beauty				Sports				Clothing			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
w/o DN	0.0395	0.0616	0.0250	0.0327	0.0244	0.0380	0.0162	0.0202	0.0189	0.0289	0.0107	0.0136
w/o IS	0.0408	0.0630	0.0252	0.0330	0.0245	0.0384	0.0165	0.0206	0.0192	0.0293	0.0109	0.0139
w/o QA-R	0.0401	0.0632	0.0245	0.0327	0.0240	0.0382	0.0160	0.0202	0.0188	0.0287	0.0103	0.0132
w/o QA-L	0.0408	0.0639	0.0248	0.0330	0.0245	0.0384	0.0165	0.0207	0.0194	0.0295	0.0106	0.0136
w/o MK	0.0413	0.0641	0.0253	0.0334	0.0256	0.0407	0.0168	0.0212	0.0196	0.0303	0.0108	0.0139
w/o RL	0.0417	0.0646	0.0255	0.0323	0.0255	0.0406	0.0166	0.0210	0.0198	0.0305	0.0110	0.0141
SeqEditor	**0.0430**†	**0.0658**†	**0.0260**†	**0.0341**†	**0.0266**†	**0.0414**†	**0.0171**†	**0.0214**†	**0.0205**†	**0.0308**†	**0.0112**†	**0.0142**†

4.3 Ablation Study

To study the contribution of each proposed technique in SeqEditor, we conduct experiments on several variants, as shown in Table 2. To ensure a consistent evaluation setting, SeqEditor and all its variants use SASRec as the recommendation model. The configurations of these variants are as follows:

- *w/o DN*: We remove both denoising operations and only perform information supplementation on input UBSs.
- *w/o IS*: We remove information supplementation and only perform denoising on input UBSs.
- *w/o QA-R*: We remove the Quality Assessor and randomly select seed UBSs for annotated data construction, instead of selecting them based on their quality scores.
- *w/o QA-L*: We replace the Quality Assessor's gradient-based scoring with loss-based scoring, selecting seed UBSs with the lowest prediction losses averaged across the same auxiliary-model checkpoints.
- *w/o MK*: We compute sequence quality scores using a single checkpoint of the auxiliary recommender, instead of averaging scores across multiple checkpoints.
- *w/o RL*: We remove RL from the optimization of the Quality Editor and use the SFT model for sequence editing.

Based on the results in Table 2, we draw the following conclusions: (i) Removing either denoising (*w/o DN*) or information supplementation (*w/o IS*) leads to a performance drop, demonstrating that both operations contribute to improving UBSs for recommendation. (ii) Replacing quality-based selection with either random selection (*w/o QA-R*) or loss-based selection (*w/o QA-L*) reduces recommendation performance. These results support the effectiveness of the Quality Assessor in identifying useful seed sequences and suggest that reference-aligned training utility is a more effective selection criterion than prediction loss alone. (iii) Using a single checkpoint (*w/o MK*) leads to performance drops. This suggests that aggregating scores across training stages can improve seed selection by accounting for variation in sequence training utility. (iv) Removing RL (*w/o RL*) also reduces recommendation performance, supporting its contribution beyond SFT to learning an effective sequence-editing policy.

4.4 Visualization Analysis

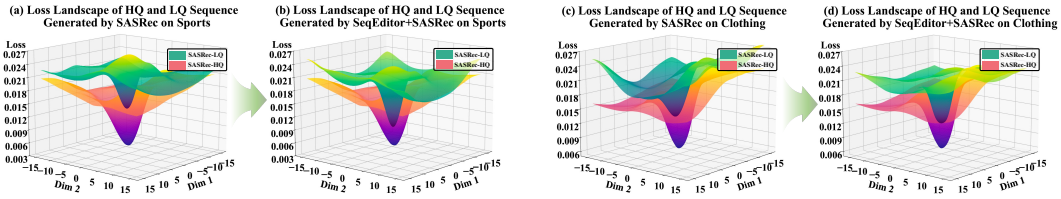

Figure 3: Comparison of loss landscapes for SASRec optimized on UBSs of different quality and edited UBSs in terms of Sports and Clothing datasets.

4.4.1 Visualization on Model's Parameter Space.

In this part, we visualizes the loss landscape (Li et al., 2018) of different models. This procedure is detailed as follows: Taking Sports dataset as example, the UBSs of this dataset are divided into high- and low-quality groups based on the UBS quality score $Q(S)$ in Eq. (6) (The division approach is detailed in Appendix B.1). Then, two SASRec are optimized separately on these two sets, denoted as SASRec-HQ and SASRec-LQ, and their loss landscapes are visualized in Figure 3-(a). In addition, we edit the UBSs in the low-quality group using SeqEditor and optimize a new SASRec on the edited UBSs. Its loss landscape is visualized in Figure 3-(b).

By comparing Figure 3-(a) and Figure 3-(b), we observe that the SASRec-LQ in Figure 3-(b) exhibits a flatter loss landscape and converges to a lower global minimum loss. A similar phenomenon can be observed on Clothing dataset, as shown in Figure 3-(c) and (d). This improvement stems from SeqEditor alleviating quality flaw problem in low-quality UBSs, thereby enhancing their quality.



Figure 4: The t-SNE visualization of user representations generated by SASRec and SeqEditor+SASRec on Sports and Clothing datasets.

4.4.2 Visualization on User Preferences.

we conduct a visual analysis to intuitively investigate the effectiveness of SeqEditor. Specifically, taking the Sports dataset as an example, we first randomly select three items. For each item, we construct a user group comprising users whose next interaction target is that item, ensuring that users within each group share similar preferences. We then obtain the representations of these users using the optimized SASRec and SeqEditor+SASRec, respectively. These representations are visualized

via t-SNE (Maaten & Hinton, 2008). The experimental results on Sports and Clothing datasets are presented in Figure 4.

Compared to SASRec, the user representations generated by SeqEditor+SASRec exhibit closer within-group distances and further between-group distances. The reason is that SeqEditor improves the quality of UBSs by mitigating the quality flaw problem, resulting in better user representations. These refined user representations lead to tighter clusters for users that target the same item and greater separation for those targeting different items.

4.5 Case Study

To investigate the interpretability of SeqEditor, we conduct a case study as illustrated in Figure 5. Specifically, we first randomly select a UBS and perform recommendation using the optimized SASRec. We observe that the target item ("Silk Tie") does not appear in the top 20 recommended items. We then edit this UBS using SeqEditor and examine how the edited sequence affects the recommendations.

During this editing process, SeqEditor drops the noise item "Ring Set" from the original UBS as well as the supplementary noise item "Bridle Belt", while retaining the meaningful supplementary item "Black Suit". When SASRec is applied again to the edited UBS, the target item is successfully included in the recommendation results, showing a clear change from the original recommendation. This case thus clearly demonstrates the effectiveness of SeqEditor.

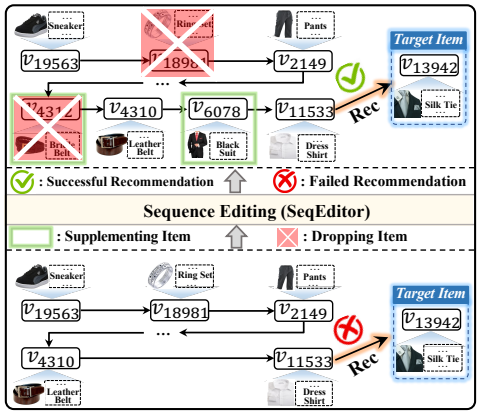

Figure 5: Case study of SeqEditor.

4.6 The SFT and RL Optimization Process

To intuitively demonstrate the optimization process of SeqEditor, we record the changes in its F1 score on the validation set during both the SFT and RL stages across three datasets. Based on the results shown in Figure 6-(a), the following observations can be summarized: (i) As the training steps increased, the performance of SeqEditor during the SFT stage improved steadily and gradually plateaued. (ii) After transitioning from the SFT stage to the RL stage, the model performance shifted from a plateau to an upward trend, demonstrating the effectiveness of the RL.

In addition, to validate the efficacy of the interest-aware reward function $R^{int}(\cdot)$, we separately employ it and the F1-score as reward functions in DAPO, optimizing two corresponding SeqEditor. We then record the recommendation performance (Recall@5) of SeqEditor + SASRec on three datasets, as shown in Figure 6-(b). Notably, $R^{int}(\cdot)$ achieves superior performance compare to the F1-score. This improvement can be attributed to its effectiveness in reducing false positives in SeqEditor's inference results.

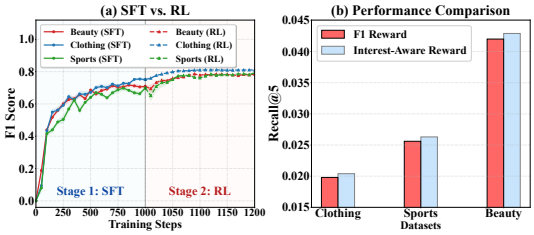

Figure 6: The performance of SeqEditor during SFT and RL stages (a), and comparison of Interest-Aware Reward function and F1-score (b).

5 Conclusion

In this paper, we propose a novel framework, called SeqEditor, to improve the quality of UBSs, thus enhancing the performance of SRSs. The pipeline of SeqEditor involves a quality assessment phase (Quality Assessor) followed by a noise removal phase (Quality Editor). Building upon the optimized SeqEditor, its inference process, i.e., Sequence Editing, effectively addresses both item missing and noise in UBSs by executing a "denoise-supplement-denoise" workflow.

LLM USAGE DISCLOSURE

Generative AI tool (GPT 5.6) were used solely for grammar checking and language polishing during manuscript preparation. They were not used to formulate the research ideas, develop the methodology, implement the code, conduct experiments, analyze the results, or draw conclusions. All AI-assisted edits were reviewed and approved by the authors, who take full responsibility for the content of the manuscript.

ETHICS STATEMENT

This work uses three publicly available Amazon Reviews benchmark datasets and processes only anonymized user–item interaction sequences and item metadata. We do not collect private user data or attempt to identify individual users, and all experimental results are reported in aggregate. Because SeqEditor may remove genuine interactions or insert plausible but unobserved items when its quality judgments are incorrect, edited sequences should be treated as model-generated training signals rather than factual records of user behavior. Real-world deployment should therefore include appropriate privacy safeguards, validation across different user groups and domains, and continuous monitoring of erroneous edits and their effects on downstream recommendations.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide an anonymous repository at `https://anonymous.4open.science/r/SeqEditor_2027` containing the implementation, data preprocessing and sequence-editing pipelines, model training and evaluation scripts, prompts, and configuration files. The main paper and appendix report the dataset statistics, quality-assessment procedure, annotated-data construction, SFT and RL objectives, hyperparameter settings, hardware configuration, evaluation metrics, and repeated-run protocol. The Amazon Reviews datasets and pretrained models used in our experiments are publicly available from their original sources. These resources support end-to-end reproduction of SeqEditor and its evaluation with different sequential recommendation models.

REFERENCES

Himan Abdollahpouri and Masoud Mansoury. Multi-sided exposure bias in recommendation. *arXiv preprint arXiv:2006.15772*, 2020.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 378–387, 2021.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600. IEEE, 2017.

Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. Exploiting llm quantization. *Advances in Neural Information Processing Systems*, 37:41709–41732, 2024.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

Yongqiang Han, Hao Wang, Kefan Wang, Likang Wu, Zhi Li, Wei Guo, Yong Liu, Defu Lian, and Enhong Chen. Efficient noise-decoupling for multi-behavior sequential recommendation. In *Proceedings of the ACM Web Conference 2024*, pp. 3297–3306, 2024.

Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 191–200. IEEE, 2016.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian Mcauley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. In *International Conference on Machine Learning*, pp. 24004–24024. PMLR, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*, 2018.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1419–1428, 2017.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Peihua Mai and Yan Pang. Vertical federated graph neural network for recommender system. In *International Conference on Machine Learning*, pp. 23516–23535. PMLR, 2023.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37:57018–57049, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 811–820, 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 565–573, 2018.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gemini Team et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 1(5):6, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Trinh Xuan Tuan and Tu Minh Phuong. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems*, pp. 138–146, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Bohao Wang, Feng Liu, Changwang Zhang, Jiawei Chen, Yudi Wu, Sheng Zhou, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, et al. Llm4dsr: Leveraging large language model for denoising sequential recommendation. *ACM Transactions on Information Systems*, 44(1):1–32, 2025.

Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 403–412, 2015.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. Sse-pt: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM conference on recommender systems*, pp. 328–337, 2020.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 346–353, 2019.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv e-prints*, pp. arXiv–2505, 2025a.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025b.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. *Advances in Neural Information Processing Systems*, 37:62279–62309, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Chi Zhang, Qilong Han, Rui Chen, Xiangyu Zhao, Peng Tang, and Hongtao Song. Ssdrec: Self-augmented sequence denoising for sequential recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 803–815. IEEE, 2024a.

Dan Zhang, Yangliao Geng, Wenwen Gong, Zhongang Qi, Zhiyu Chen, Xing Tang, Ying Shan, Yuxiao Dong, and Jie Tang. Recdcl: Dual contrastive learning for recommendation. In *Proceedings of the ACM Web Conference 2024*, pp. 3655–3666, 2024b.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 2024.

Peilin Zhou, Jingqi Gao, Yueqi Xie, Qichen Ye, Yining Hua, Jaeboum Kim, Shoujin Wang, and Sunghun Kim. Equivariant contrastive learning for sequential recommendation. In *Proceedings of the 17th ACM conference on recommender systems*, pp. 129–140, 2023.

Pablo Zivic, Hernan Vazquez, and Jorge Sánchez. Scaling sequential recommendation models with transformers. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1567–1577, 2024.

Appendix

- **A. Related Work** 14
- **B. Supplementary Methods** 15
  - B.1 *Quality-based Data Division for Empirical Studies* 15
  - B.2 *Training Sample Cases for SFT and RL in SeqEditor* 15
  - B.3 *The Analysis of Reward Functions for RL* 18
  - B.4 *Parameters Setting and Algorithm of SeqEditor* 15
- **C. Supplementary Experiments** 16
  - C.1 *Dataset Description* 16
  - C.2 *Recommendation Models* 17
  - C.3 *Visualization on Model's Parameter Space* 17
  - C.4 *The Cross-Domain Transferability of SeqEditor* 17
  - C.5 *The Impact of Model Scale and Complexity on SeqEditor* 18
  - C.6 *The Analysis of Reward Functions for RL* 18
  - C.7 *The Correlation between UBS Quality and Model Performance* 19
  - C.8 *Performance Comparison with Denoising-based Baselines* 19
  - C.9 *The Impact of LLM Embedding* 19
  - C.10 *Hyperparameter Analysis* 20

A Related Work

Sequential Recommendation. Sequential Recommendation captures evolving user preferences by modeling temporal dependencies in user behavior sequences. Early methods use Markov Chain to model the sequential patterns (Rendle et al., 2010; He & McAuley, 2016; Wang et al., 2015). The rise of deep learning has shifted the paradigm towards neural sequential models. Initially, RNN-based methods (Hidasi et al., 2015; Li et al., 2017) become the mainstream approach, successfully unlocking the ability to model dynamic user preferences over time. Subsequently, researchers explore alternative architectures, including CNN-based (Tang & Wang, 2018; Tuan & Phuong, 2017) and GNN-based (Wu et al., 2019; Chang et al., 2021; Mai & Pang, 2023) studies, to model complex item transitions. To better capture long-range dependencies, the Transformer-based efforts (Kang & McAuley, 2018; Sun et al., 2019; Wu et al., 2020; Zivic et al., 2024) have rapidly advanced. For example, SASRec (Kang & McAuley, 2018) adapts the self-attention mechanism for sequential recommendation, thereby identifying relevant historical items. SRT (Zivic et al., 2024) modifies the Transformer through a feature extractor and contrastive learning to decouple its parameters from the scale of item set. Despite the dominance of these discriminative baselines, a new wave of research explores a generative paradigm (Rajput et al., 2023; Zhai et al., 2024; Zheng et al., 2024; Hou et al., 2025). TIGER (Rajput et al., 2023) introduces the concept of semantic IDs to facilitate generative retrieval. HSTU (Zhai et al., 2024) reformulates recommendation as a generative sequential transduction task and proposes an architecture that scales efficiently to long user sequences.

SFT and RL for LLMs. LLMs have revolutionized NLP via the Transformer and scaling laws (Vaswani et al., 2017; Kaplan et al., 2020; Hägele et al., 2024). Although proprietary models like GPT (Achiam et al., 2023) and Gemini (Team et al., 2023; 2025) lead the field, highly capable open-source counterparts (Touvron et al., 2023; Liu et al., 2024; Yang et al., 2025b) facilitate broader research. For instance, DeepSeek (Liu et al., 2024) optimizes inference via Mixture-of-Experts (MoE), while Qwen3 (Yang et al., 2025b) enhances reasoning through large-scale data synthesis. To adapt these models for downstream tasks via SFT (Wei et al., 2021), the associated heavy memory costs are mitigated using LoRA (Hu et al., 2022), layerwise-freezing (Pan et al., 2024), and quantization technologies (Ashkboos et al., 2024; Egashira et al., 2024). Beyond SFT, human preference alignment requires RL. Alignment strategies evolved from PPO (Schulman et al., 2017) to efficient alternatives (Rafailov et al., 2023): GRPO (Shao et al., 2024) eliminates the value network, while DAPO (Yu et al., 2025) improves convergence. To mitigate reward over-optimization, recent reward models leverage hidden-state regularization (Yang et al., 2024), reasoning-chain evaluation (Chen et al., 2025), and process reward models (Setlur et al., 2024).

B SUPPLEMENTARY METHODS

B.1 QUALITY-BASED DATA DIVISION FOR EMPIRICAL STUDIES

This subsection supplements the method for partitioning training UBSs into high- and low-quality sets, which is employed in Sec. 1 and Sec. 4.4. Specifically, given a UBS S, we obtain its quality score $Q(S)$ via Eq. (6). All UBSs in the training set $\mathcal{D}$ are then sorted in descending order of $Q(S)$. The top 20% constitute the high-quality set $\mathcal{D}_H$, and the bottom 20% form the low-quality set $\mathcal{D}_L$. Let $\{S^{(i)}\}_{i=1}^{N}$ denote the sorted UBSs. This procedure is formally defined as follows:

$$\mathcal{D}_H = \{S^{(i)} \mid 1 \le i \le \lfloor 0.2N \rfloor\}, \qquad \mathcal{D}_L = \{S^{(i)} \mid N - \lfloor 0.2N \rfloor + 1 \le i \le N\}. \tag{18}$$

SASRec is then trained separately on these two sets, yielding SASRec-HQ and SASRec-LQ. The loss landscape for each model is then computed. In addition, we randomly sample 20% of the remaining UBSs as a test set to evaluate the recommendation performance of the two models.

B.2 TRAINING SAMPLE CASES FOR SFT AND RL IN SEQEDITOR

SeqEditor is optimized in two stages: SFT and RL. Figure 7 illustrates the construction of training data for each stage. As shown in Figure 7-(a), each sample in the SFT stage primarily consists of text descriptions and category information for each item within UBS. Its label includes the number of noisy items and their corresponding IDs. In addition, as presented in Figure 7-(b), the RL training samples have a similar structure. The key distinction is that their labels additionally include user interest count information, which is used to compute the interest-aware reward $R^{int}(\cdot)$ (Eq. (9)).

(a) SFT Training Example on Clothing Dataset

System Prompt: You are a useful and harmless assistant

User Prompt: Here is the sequence of items that user interacted. Each item consists of an ID, description and category:

<<item 17472: State O Maine - Mens Fleece Pajama Lounge Pant, Navy 29651-Small; categories: [['Clothing, Shoes & Jewelry', 'Men', 'Clothing', 'Sleep & Lounge', 'Sleep Bottoms']]>>
<<item 11: Suunto Vector Wristop Computer with Compass, Altimeter & Barometer; categories: [['Electronics', 'GPS & Navigation', 'Sports & Handheld GPS', 'Handheld GPS Units'], ['Clothing, Shoes & Jewelry', 'Women', 'Watches', 'Wrist Watches']]>>
…
<<item 15364: Dirty Laundry Women's Showstopper Motorcycle Boot; categories: [['Clothing, Shoes & Jewelry', 'Shoes & Accessories: International Shipping Available'], ['Clothing, Shoes & Jewelry', 'D', 'Dirty Laundry']]>>
…

Based on the sequence of items the user has clicked, analyze the order of these items and their titles to determine if there are any noise items in the user's behavior sequence. Based on your analysis, return the IDs of the noise items

Label: There are #2# noised item(s): <<17472, 15364>>

(b) RL Training Example on Clothing Dataset

System Prompt: You are a useful and harmless assistant

User Prompt: Here is the sequence of items that user interacted. Each item consists of an ID, description and category:

<<item 22646: Qs 11 Pieces Baby's Headbands Girl's Chiffon Headband Hair Flower (11 Pack); categories: [['Clothing, Shoes & Jewelry', 'Novelty, Costumes & More', 'Costumes & Accessories']]>> <<item 4580: ROLO1MM Italian Sterling Silver 1mm Rolo Sturdy 14, 16, 18, 20, 22, 24, 30 Inch Chain Necklace; categories: [['Clothing, Shoes & Jewelry', 'Women'], ['Clothing, Shoes & Jewelry', 'Novelty, Costumes & More', 'Novelty', 'Jewelry', 'Necklaces & Pendants']]>>
…
<<item 21744: Clarks Women's Roya Vanna Wedge Sandal; categories: [['Clothing, Shoes & Jewelry', 'Shoes & Accessories: International Shipping Available'], ['Clothing, Shoes & Jewelry', 'C', 'Clarks']]>>
…

Based on the sequence of items the user has clicked, analyze the order of these items and their titles to determine if there are any noise items in the user's behavior sequence. Based on your analysis, return the IDs of the noise items.

Label: There are #2# noised item(s): <<17472, 15364>>; Interest number: 4

Figure 7: Training Sample Cases for SFT and RL Stages on Clothing Dataset.

Table 3: The training settings of SFT and RL.

SFT Setting		**RL Setting**		**RL Setting**	
Parameter Name	Parameter Setting	Parameter Name	Parameter Setting	Parameter Name	Parameter Setting
Quantization Bits	8	Quantization Bits	8	Learning Rate	5e-6
Precision	bfloat16	Precision	bfloat16	Weight Decay	0.1
LoRA Rank	8	LoRA Rank	8	Gradient Clipping	1.0
LoRA Alpha	32	LoRA Alpha	32	Mini-Batch Size	16
Learning Rate	5e-5	Parallel Strategy	FSDP + vLLM	Samples Per Prompt	8
Global Batch Size	2	Max Prompt Length	8192	Temperature	1.2
Grad Accumulation	8	Max Response Length	4,096	Clip Low Ratio	0.2
Max Sequence Length	8000	Total Context Window	12,288	Clip High Ratio	0.4

B.3 PARAMETERS SETTING AND ALGORITHM OF SEQEDITOR

The implementation of SeqEditor is based on PyTorch. During the SFT stage, SeqEditor is fine-tuned with LoRA (Hu et al., 2022), with key parameters set to their default values as suggested in the

Algorithm 1 Implementation of SeqEditor

Input: UBS data $\mathcal{D}_{\text{tr}} = \{S^{(i)}\}_{i=1}^{N}$; SeqEditor $\mathcal{F}_{\Theta_{SE}}(\cdot)$ and recommendation model $\mathcal{M}_{\Theta_{Rec}}(\cdot)$;
Output: Optimized parameters Θ_{SE} and Θ_{Rec};
Procedure:
Sample a length-stratified reference set $\mathcal{R}$ from $\mathcal{D}_{\text{tr}}$; train auxiliary SASRec on $\mathcal{D} = \mathcal{D}_{\text{tr}} \setminus \mathcal{R}$ and save C checkpoints;
Compute $Q(S)$ by averaging shared-encoder gradient alignment scores across the fixed checkpoints (Eqs. (2)–(4) and (6));
Select the high-quality UBS set $\mathcal{D}_H = \{S \in \mathcal{D} : Q(S) > 0\}$;
Build the annotated training set $\mathcal{D}_A$ by adding noise to $\mathcal{D}_H$ (Eq. (7));
#SFT Stage:
  for every epoch **do**
    Optimize Θ_{SE} by minimizing $\mathcal{L}_{SFT}$ on $\mathcal{D}_A$ (Eq. (8));
    Apply early stopping;
  end for
#RL Stage:
  for every epoch **do**
    Optimize Θ_{SE} using DAPO (Eqs. (10)–(11)) with the interest-aware reward (Eq. (9));
    Apply early stopping;
  end for
Edit each $S \in \mathcal{D}_{\text{tr}}$ through denoising, supplementation, and denoising (Eqs. (12)–(17)), obtaining $\widetilde{S}$ for recommendation training;
#Optimizing Recommendation Model:
  for every epoch **do**
    Optimize Θ_{Rec} by minimizing the recommendation loss on the edited UBSs;
    Apply early stopping;
  end for

original paper, including LoRA rank: 8, LoRA alpha: 32, and quantization bits: 8. In the RL stage, SeqEditor is optimized using DAPO (Yu et al., 2025), with main parameters set conventionally as clip low ratio: 0.2, clip high ratio: 0.4, overlong penalty factor: 1.0, and number of responses per prompt: 8. For the GPU configuration, all experiments are conducted on NVIDIA 4 * A800 (80GB). More detailed training configurations are provided in Table 3. Additionally, the hyperparameter η in Eq. (17) is set to 0.8, and the max number of noise items $K = 5$. The number of checkpoints C in Eq. (6) is set to 4. The ordered adjacent-pair counts and their marginal counts of items are counted based on the training set.

The algorithm of SeqEditor mainly comprises four steps: (i) Evaluating UBSs quality via the Quality Assessor to extract high-quality subset, thus building annotated training data; (ii) Optimizing SeqEditor through SFT; (iii) Optimizing SeqEditor through RL; (iv) Applying the optimized SeqEditor to edit UBSs. These edited UBSs are leveraged to optimize various SRSs. The complete procedure is described as Algorithm 1.

Table 4: Statistics of the three experimental datasets.

Dataset	#User	#Item	#Interaction	#Avg.t
Beauty	22,363	12,101	176,139	8.87
Sports	35,598	18,357	260,739	8.32
Clothing	39,387	23,033	278,677	7.07

C Supplementary Experiments

C.1 Dataset Description

To validate the effectiveness of SeqEditor across different recommendation scenarios, we select three types of benchmark datasets from Amazon Review, including (i) Beauty; (ii) Sports and Outdoors; and (iii) Clothing, Shoes and Jewelry. For ease of reference, these are referred to as *Beauty*, *Sports*, and *Clothing*, respectively. Detailed dataset statistics are presented in Table 4.

C.2 Recommendation Models

To validate the general efficacy of SeqEditor, we conduct performance testing and comparison using both the original UBSs and edited UBSs by SeqEditor across two types of SRSs, including discriminative recommendation (DR)-based models and generative recommendation (GR)-based models. Detailed descriptions of these models are provided below.

DR-based Models.

- **GRU4Rec** (Hidasi et al., 2015): This model applies Gated Recurrent Unit (GRU) networks (Dey & Salem, 2017) to capture temporal dependencies in user interaction sequences.
- **SASRec** (Kang & McAuley, 2018): It employs a Transformer-based architecture with self-attention to dynamically weigh all previous interactions, effectively capturing both long- and short-term dependencies for next-item prediction.
- **BERT4Rec** (Sun et al., 2019): It extends the BERT framework (Devlin et al., 2019) to recommendation, this model employs bidirectional self-attention and masked item prediction to learn contextualized sequence representations

GR-based Models.

- **TIGER** (Rajput et al., 2023): It introduces a Semantic ID scheme to represent items as discrete codes, and uses a bidirectional Transformer to generate these codes autoregressively for recommendation.
- **HSTU** (Zhai et al., 2024): This model combines Transformers in a hierarchical architecture to efficiently model both long-range dependencies and complex patterns in user sequences.

C.3 Visualization on Model's Parameter Space

To further investigate the influence of SeqEditor on the parameter space of recommendation models, we extend the experiments in Sec. 4.4 on the Baby datasets. The results are shown in Figure 8, which are consistent with the findings in Sec. 4.4. Compared with the SASRec optimized on the original low-quality data, the SASRec optimized with SeqEditor-edited low-quality data exhibits a flatter loss landscape and converges to a lower global minimum loss. This further corroborates the effectiveness of SeqEditor from a parameter-space perspective.

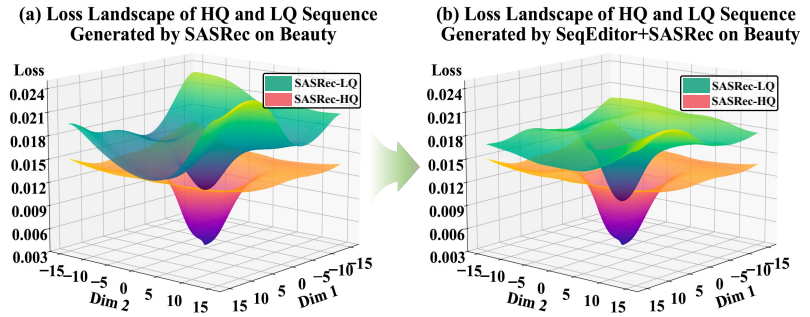

Figure 8: Comparison of loss landscapes for SASRec optimized on UBSs of different quality and edited UBSs in terms of Sports and Clothing datasets.

C.4 The Cross-Domain Transferability of SeqEditor

In this subsection, we validate the cross-domain transferability of SeqEditor based on the SASRec. This includes the mixed training of SeqEditor on multiple source-domain datasets and the evaluation of its cross-domain performance on a target domain dataset. The experimental results are shown in Table 5. Compared to "Self $\rightarrow$ Self", we observe that "Self + X $\rightarrow$ Self" yields better performance, where "Self" refers to any single domain among Clothing, Sports, or Beauty, and "X" denotes a combination of the other domains. This indicates the presence of cross-domain correlations among item texts in recommendation—particularly between Clothing and Sports—and validates the effective cross-domain transfer ability of SeqEditor.

Table 5: The cross-domain transferability validation for SeqEditor on more configurations, where SD and TD denote the source and target domain, respectively.

SD → TD	R@10	N@10	SD → TD	R@10	N@10	SD → TD	R@10	N@10
w/o SeqEditor	0.0605	0.0318	w/o SeqEditor	0.0350	0.0192	w/o SeqEditor	0.0272	0.0130
Beauty → Beauty	0.0658	0.0341	Sports → Sports	0.0414	0.0214	Clothing → Clothing	0.0308	0.0142
Sports → Beauty	0.0643	0.0334	Beauty → Sports	0.0392	0.0204	Sports → Clothing	0.0304	0.0136
Clothing → Beauty	0.0640	0.0338	Clothing → Sports	0.0407	0.0208	Beauty → Clothing	0.0292	0.0135
Clothing + Beauty → Beauty	0.0661	0.0345	Clothing + Beauty → Sports	0.0406	0.0207	Clothing + Beauty → Clothing	0.0311	0.0140
Sports+ Beauty → Beauty	0.0660	0.0342	Sports+ Beauty → Sports	0.0415	0.0212	Sports+ Beauty → Clothing	0.0300	0.0132
Clothing + Sports → Beauty	0.0625	0.0319	Clothing + Sports → Sports	0.0418	0.0214	Clothing + Sports → Clothing	0.0314	0.0142
Clothing + Sports + Beauty → Beauty	**0.0662**	**0.0346**	Clothing + Sports + Beauty → Sports	**0.0418**	**0.0215**	Clothing + Sports + Beauty → Clothing	**0.0315**	**0.0142**

C.5 The Impact of Model Scale and Complexity on SeqEditor

To evaluate the impact of model scale on the training cost and inference efficiency of SeqEditor, we conduct experiments using Qwen-series models of varying parameter sizes, including 4B, 8B, 14B, and 32B. We record their time efficiency and GPU costs during both training and inference stages, and compare their effects on the recommendation performance of SeqEditor+SASRec.

The experimental results are shown in Table 6. During training, both time and GPU costs increased notably with model size, particularly in the RL stage. Inference is similarly affected by model scale, with larger models leading to slower response times and higher GPU costs. In terms of recommendation performance, experiments indicate that the performance of SeqEditor+SASRec does not improve notably with increased model scale of SeqEditor. The reason is that the task of SeqEditor is relatively simple. Smaller models already perform adequately while maintaining better efficiency.

Table 6: Impact of SeqEditor scale on time efficiency, GPU cost, and SASRec performance in Clothing dataset.

LLM Size	Training SFT (LoRA)		Training RL (LoRA)		Inference		SASRec Performance	
	Time/Epoch (h)	GPU Costs (GB)	Time/Epoch (h)	GPU Costs (GB)	Time/Sample (s)	GPU Costs (GB)	R@10	N@10
Qwen3-4B	2.76	62.00	4.18	145.72	0.045	16.47	0.0308	0.0138
Qwen3-8B	5.53	67.58	5.91	158.09	0.083	32.35	0.0311	0.0140
Qwen3-14B	8.60	75.39	12.32	176.36	0.112	48.21	0.0313	0.0141
Qwen3-32B	15.34	78.25	26.27	307.70	0.873	71.99	0.0314	0.0141

C.6 The Analysis of Reward Functions for RL

To further investigate the impact of interest-aware reward function $R^{int}(\cdot)$ on reinforcement learning, relevant experiments are performed in this subsection. First, the test UBSs containing noise items are divided into four groups according to the increasing number of user interests. In the RL stage, we train two SeqEditor, one using F1 and the other using interest-aware reward as the reward function. The two models are then applied to denoise the UBSs within each group for precision calculation.

The experimental results are shown in Figure 9. On the Sports and Clothing datasets, the performance of both models declines as the number of user interests increases. The Beauty dataset generally follows a similar trend, with the exception of the "Other" group. This discrepancy may be attributed to the relatively small size of the Beauty dataset, where the "Other" group accounts for only 11.75%

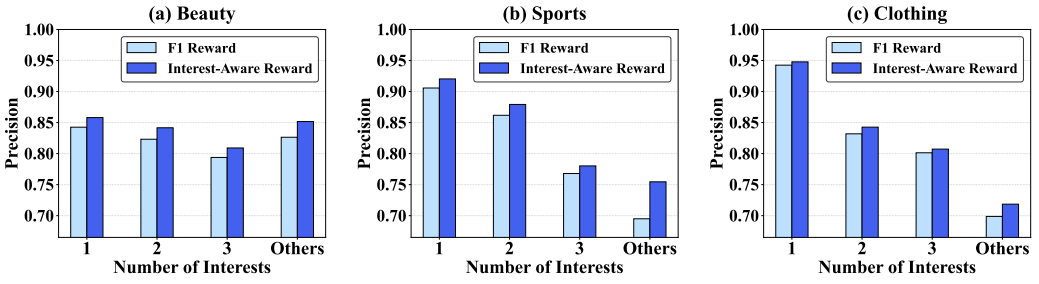

Figure 9: Performance comparison of SeqEditor Using F1 and Interest-Aware Reward as reward function in RL stage across different interest groups.

of the total testing UBSs, leading to biased precision for this group. Additionally, compared with F1,

interest-aware reward function is found to significantly improve the denoising performance of the model on UBSs with a large number of user interests. This is mainly due to the higher sensitivity of this reward to users with complex interests. By penalizing false positives more heavily, the model is guided to reduce the erroneous removal of normal items.

C.7 The Correlation between UBS Quality and Model Performance

In this subsection, we investigate the relationship between UBSs quality change and the performance change of SRSs. Concretely, given a UBS S and its edited version $\widetilde{S}$ obtained via SeqEditor, we compute their quality scores $Q(S)$ and $Q(\widetilde{S})$ through Eq. (6). Thereafter, we compute the quality score change $\Delta Q(S) = Q(\widetilde{S}) - Q(S)$ for each UBS. In addition, we train two SASRec separately based on $\widetilde{S}$ and S, and evaluate their performance ($Recall@10$) on the same test set. The difference in their performance is denoted as $\Delta Recall@10$. The experimental results are shown in Figure 10. We observe a positive correlation between quality change and performance change. This indicates that SeqEditor enhances recommendation performance of SRSs by improving UBS quality.

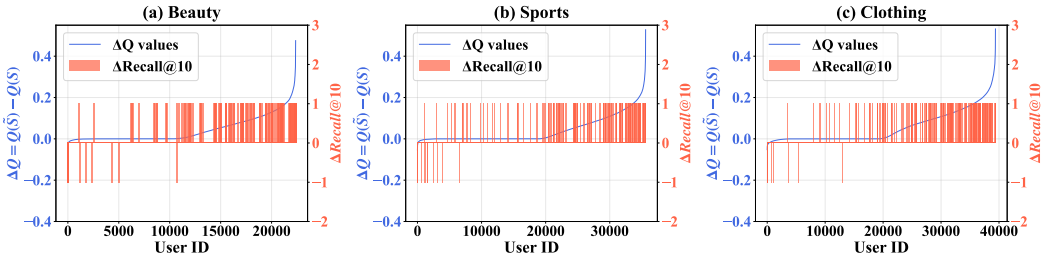

Figure 10: The correlation between changes in UBS quality and SRSRec performance.

C.8 Performance Comparison with Denoising-based Baselines

In this subsection, we compare SeqEditor (with SASRec as the backbone) with recent denoising-based models (Zhou et al., 2023; Zhang et al., 2024a; Wang et al., 2025) in terms of performance. These denoising-based models are introduced as follows:

- **ECL-SR** (Zhou et al., 2023): It proposes equivariant contrastive learning for sequential recommendation, making representations sensitive to invasive augmentations while insensitive to mild ones.
- **SSDREC** (Zhang et al., 2024a): This framework adopts a three-stage paradigm: learning inter-sequence relations as priors, self-augmenting sequences, and hierarchically denoising to avoid false augmentations.
- **LLM4DSR** (Wang et al., 2025): It designs a self-supervised fine-tuning task to activate the LLM's ability to identify noisy items and suggest replacements, and incorporates an uncertainty estimation module to retain only high-confidence predictions.

The experimental results shown in Table 7 demonstrate the clear superiority of SeqEditor. Compared with a recent LLM-based method (LLM4SDR), SeqEditor offers the following advantages: (i) Equipped with Quality Assessor, SeqEditor constructs higher-quality training signals for the LLM by explicitly modeling noise in user behavior sequences. (ii) SeqEditor incorporates user interest–aware RL, making it more robust in scenarios where user preferences evolve dynamically. (iii) Instead of generating item descriptions in natural language and matching missing items across the full item space, SeqEditor directly identifies missing items based on the mutual information and LLM verification, enabling more accurate and efficient missing supplementation.

C.9 The Impact of LLM Embedding

To investigate whether SeqEditor's performance gains stem from feature engineering or from sequence editing, we directly feed the raw user behavior sequence (including item text descriptions) into

Table 7: Performance comparison with denoising-based models.

Model	Beauty				Sports				Clothing			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
ECL-SR	0.0407	0.0621	0.0242	0.0323	0.0230	0.0363	0.0159	0.0194	0.0179	0.0278	0.0105	0.0132
SSDREC	0.0409	0.0625	0.0243	0.0326	0.0239	0.0378	0.0162	0.0199	0.0183	0.0284	0.0106	0.0134
LLM4SDR	0.0414	0.0630	0.0245	0.0329	0.0245	0.0382	0.0163	0.0201	0.0188	0.0287	0.0109	0.0135
SeqEditor	**0.0430**	**0.0658**	**0.0260**	**0.0341**	**0.0266**	**0.0414**	**0.0171**	**0.0214**	**0.0205**	**0.0308**	**0.0112**	**0.0142**

the Qwen-4B model. We extract LLM-based item representations from its embedding layer and use them as input features for *SASRec*, without performing any sequence editing. This variant is denoted as *SASRec+LLM Embedding*. We then compare its performance with the *SASRec*, *SASRec+SeqEditor*, and *SASRec+SeqEditor+LLM Embedding*. The results are shown in the Table 8. We observe that *SASRec+LLM Embedding* underperforms *SASRec+SeqEditor*, indicating that editing user behavior sequences provides additional gains beyond mere feature enhancement. Importantly, *SASRec+SeqEditor+LLM Embedding* achieves the best performance, showing that LLM feature representation and the sequence editing mechanism contribute jointly from two complementary perspectives.

Table 8: Impact of LLM embedding on the performance of different models.

Variant Model	Beauty				Sports				Clothing			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
SASRec	0.0387	0.0605	0.0249	0.0318	0.0233	0.0350	0.0154	0.0192	0.0187	0.0272	0.0103	0.0130
SASRec+LLM Embedding	0.0401	0.0620	0.0251	0.0323	0.0240	0.0368	0.0160	0.0197	0.0193	0.0285	0.0108	0.0134
SASRec+SeqEditor	0.0430	0.0658	0.0260	0.0341	0.0266	0.0414	0.0171	0.0214	0.0205	0.0308	0.0112	0.0142
SASRec+SeqEditor+LLM Embedding	**0.0435**	**0.0667**	**0.0262**	**0.0347**	**0.0269**	**0.0421**	**0.0173**	**0.0216**	**0.0211**	**0.0317**	**0.0113**	**0.0145**

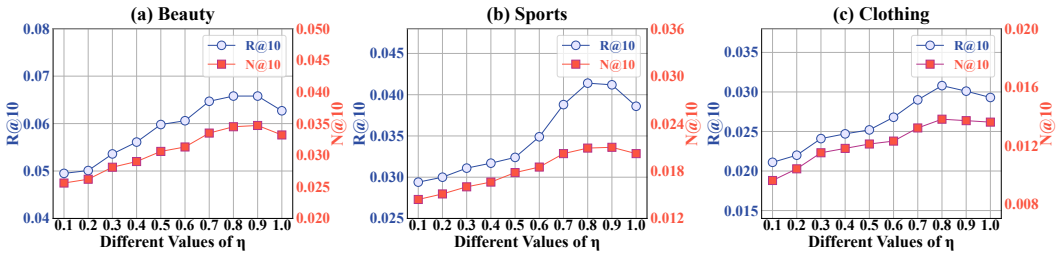

Figure 11: Hyperparameter analysis on η across three datasets.

C.10 Hyperparameter Analysis

In this subsection, we perform a hyperparameter analysis on the information supplementation threshold η (Eq. (17)). Specifically, we adopt SASRec as the recommendation model and vary η within the range of 0.1 to 1 for sequence editing. Based on these edited UBSs, we train SASRec and evaluate its performance. The experimental results are shown in Figure 11. We observe that as η increases, the model performance improves accordingly, reaching its peak around $\eta = 0.8$. This is because an overly small η may introduce noisy information into UBSs, whereas an excessively large η could lead to insufficient information supplementation.