Stablizing Recommendations by Rank-Preserving Fine-Tuning

Sejoon Oh
soh337@gatech.edu
Georgia Institute of Technology
United States

Berk Ustun
berk@ucsd.edu
University of California San Diego
United States

Julian McAuley
jmcauley@eng.ucsd.edu
University of California San Diego
United States

Srijan Kumar
srijan@gatech.edu
Georgia Institute of Technology
United States

ABSTRACT

Modern recommender systems may generate significantly different recommendations due to small perturbations in the training data. Changes in the data from one user can alter the recommendations for other unrelated users. We propose a method to stabilize recommender systems against such perturbations. This is a challenging task due to (1) the unavailability of "ideal" ranked recommendation lists; (2) the scalability of optimizing the stability of rank lists containing all items and for all training instances; and (3) the possibility of various noisy perturbations. Our method, FINEST, overcomes these challenges by first obtaining reference rank lists from a given recommendation model and then *fine-tuning* the model under simulated perturbation scenarios with rank-preserving regularization on sampled items. Our experiments on three real-world datasets demonstrate that FINEST can ensure that recommender models produce stable recommendations under a wide range of different perturbations while preserving next-item prediction accuracy.

1 INTRODUCTION

Sequential recommender systems [9, 16, 24, 30, 45] use historical user-item interactions as training data and produce ranked recommendation lists for users using the trained model and observed interactions. While existing recommenders have primarily focused on improving accuracy, there has been a surge of interest in addressing address criteria such as fairness [12, 53], diversity [7, 42], and robustness [10, 36, 44, 55, 64].

Recent work [36] has shown that sequential recommenders are unstable when faced with perturbations in the training data. Perturbations refer to changes in the data, such as the insertion, deletion, or replacement of user-item interactions. These perturbations can occur due to arbitrary noise in user activity (e.g., when a user clicks on an item multiple times) or as a result of the presence of adversaries [6, 8, 10, 33, 36, 44, 50, 55, 64, 65].

The stability of a model is defined as its ability to generate consistent recommendation lists even in the presence of noisy perturbations in the input data. To illustrate this, consider Fig. 1, which showcases two datasets: the original dataset with no perturbations (top left) and the dataset with random minor perturbations (bottom left). If a typical recommender model is trained using these two datasets, it will generate drastically different rank lists for each user (top right and middle right rows), despite the minor differences in the datasets. Our goal is to devise a stable recommender model (bottom right row) that generates rank lists similar to the original rank lists, despite the presence of perturbations.

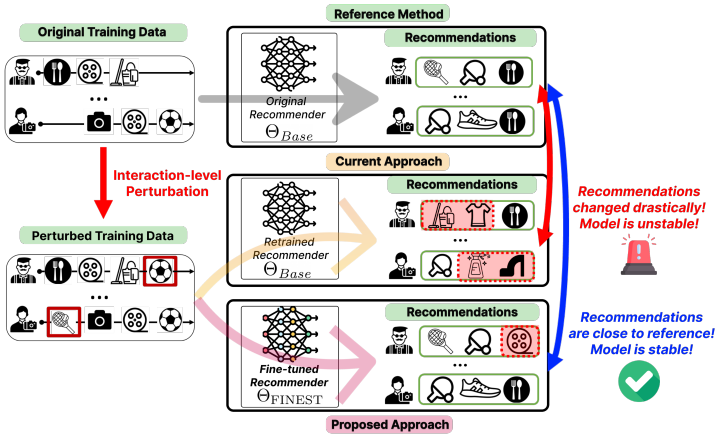


Figure 1: Existing sequential recommendation models can generate drastically different rank lists filled with irrelevant items given interaction-level training data perturbations (mid), compared to reference rank lists (top). The recommendations can be stabilized with our proposed fine-tuning method FINEST (bottom).

This instability of recommender systems can be detrimental to diverse online platforms, including high-stakes applications such as healthcare, education, and housing [47, 51, 62, 66]. For instance, minor noisy perturbations in the training data for one user can lead to drastic changes in recommendations for *all* users [36]. These changes include the top-K recommendations to users being filled with irrelevant items, which can damage the reliability of recommender systems and consequently lower user engagement and satisfaction on the platforms [23, 39]. Moreover, some user groups' recommendations can be altered more than other groups [36]. In extreme cases, an adversary can even intentionally lower the model stability by manipulating the training data, which can amplify users' mistrust and dissatisfaction with platforms.

Despite the importance of recommendation stability, there is limited research on training mechanisms that enhance the stability of recommender systems. Existing methods [18, 48, 59] for enhancing model robustness typically aim to preserve the overall accuracy metric against input perturbations. In other words, they stabilize the rank of one specific item (typically, the ground-truth next item) in a rank list rather than all the items in the list [3, 56]. Therefore,

RecSys FAccTRec Workshop '23, September 18-22, 2023, Singapore
2023. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

it is crucial to develop a technique for stabilizing recommenders (including those in deployment) against perturbations.

We propose adopting a learning-based method to stabilize rank lists in sequential recommender systems. However, there are four challenges to overcome. First, to generate consistent rank lists with and without perturbations, the model requires "*reference*" rank lists that can be used to generate stable rank lists. However, in the datasets, only ground-truth next items are present, but the desired rank lists are not. Second, there is a lack of objective functions that can optimize pairwise rank list similarity based on item order. Third, each rank list includes all the items, and each interaction generates a new rank list. Optimizing over all of these rank lists is computationally prohibitive. Finally, the interaction-level perturbations can vary, possibly introducing noise through injection, deletion, or replacement [36, 44, 64]. Since the specific type of noise is unknown before testing, it is desirable to develop an algorithm that remains agnostic to the noise during testing.

We present a fine-tuning method for sequential recommender systems that maximizes model stability while preserving prediction performance, named **FINEST** (FINE-tuning for STable Recommendations). FINEST stabilizes a pre-trained recommendation model, Θ_{Base}. Using Θ_{Base}, FINEST first obtains recommendation lists (referred to as "reference" rank lists) for all training instances (addressing challenge 1). Then, FINEST fine-tunes Θ_{Base}'s parameters to enhance stability in two steps during every epoch. Firstly, FINEST simulates a perturbation scenario by randomly sampling and perturbing a small number of interactions (e.g., 0.1%) in each epoch. This generates perturbed training data to be used in that epoch. Secondly, FINEST designs a *novel regularization function* to encourage the rank lists generated by the model (being fine-tuned with the perturbed data) to be the same as the reference rank lists. This regularization function is used along with the next-item prediction objective (addressing challenge 2). Since conserving the ranks of all items is computationally prohibitive, optimization is only performed on the top-K items (addressing challenge 3). This fine-tuned model is used during test time as-is, regardless of various perturbations during testing (addressing challenge 4).

FINEST is *model-agnostic*, meaning that it can stabilize the recommendations for *any* existing sequential recommender system against perturbations. FINEST is *a fine-tuning method* enabling it to be applied to any *pre-trained and even deployed* recommender systems without requiring training from scratch.

Experiments are conducted on three recommender models against input perturbations on three datasets. To measure the stability, we employ two ranklist stability metrics, as defined in existing works [36]: Rank-biased Overlap (RBO) [54] and Top-K Jaccard Similarity [36]. These metrics capture the differences between the rank lists with and without perturbations. With FINEST, recommender models demonstrate significantly increased stability, for example, at least 11% and 7% improvement in RBO, and 43% and 19% improvement in Jaccard score compared to the original model and baselines, respectively, while preserving model prediction accuracy. FINEST also *empirically preserves recommendation performance* due to joint training of the next-item prediction objective and the rank-preserving objective.

Table 1: Overview of existing algorithms to build robust recommender systems.

	FINEST	APT [56]	AMR [48]	APR [18]	AdvIR [38]	ACAE [59]
Enhance Model Robustness	✓	✓	✓	✓	✓	✓
Fine-tuning Applicability	✓	✓	✓	✓		✓
Sequential Recommendation	✓					✓
Simulate Perturbations in Training	✓	✓	✓			
Preserve All Users' Rank Lists	✓					

2 RELATED WORK

Robust Recommender Systems. The majority of existing training or fine-tuning methods [2–4, 11, 18, 38, 48, 56, 59] for robust recommender systems are designed to provide accurate next-item predictions in the presence of input perturbations. However, as shown in Table 1, most of these methods [18, 38, 48, 56, 59] have limitations in enhancing the ranking stability of sequential recommenders against input perturbations, as they are not optimized to preserve entire rank lists (but rather focus on ground-truth next-items) [18, 56, 59] or cannot be applied to sequential settings [38] (which can predict users' interests based on sequences of their recent interactions). A few of them [3, 48] also require additional input like images. While a few ranking-distillation methods [49, 63] can be adapted to our setting, they are unsuitable for preserving rank lists against input perturbations, as they may sacrifice the next-item prediction accuracy to achieve their goal.

Adversarial Machine Learning. Adversarial training has been widely used in computer vision and natural language processing (NLP) [14, 34, 35] areas to enhance the robustness of deep learning models. Many adversarial training methods use min-max optimization, which minimizes the maximal adversarial loss (i.e., worst-case scenario) computed with adversarial examples [52]. In computer vision, adversarial examples are generated by the Fast Gradient Sign Method [14], Projected Gradient Descent [5], or GANs [43], which can change the classification results. In the NLP area, adversarial examples are created in various ways, such as replacing characters or words in the input text and applying noise to input token embeddings [34, 35]. However, these models cannot be directly applied to recommender systems as they do not work on sequential interaction data or do not generate rank lists of items.

3 PRELIMINARIES

Sequential Recommendations. We focus on sequential recommender models in this work, which are trained to accurately predict the next item of a user based on their previous interactions. Formally, we have a set of users $\mathcal{U}$ and items $\mathcal{I}$. For a user u, their interactions are represented as a sequence of items (sorted by timestamps) denoted as $S^u = \{S_1^u, \dots S_{m^u}^u\}$, and the corresponding timestamps as $T^u = \{T_1^u, \dots T_{m^u}^u\}$. Here, $S_t^u \in \mathcal{I}$, and m^u represents the total number of interactions for a user u. We train the sequential recommender with the following loss function to predict the next item S_{t+1}^u accurately for each user u, given an item sequence

$\{S_1^u, \ldots, S_t^u\}, \forall t \in [1, m^u - 1]$.

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \sum_{t=1}^{m^u - 1} CE(\mathbf{1}^{S_{t+1}^u}, \Theta(\{S_1^u, \ldots, S_t^u\})). \quad (1)$$

Note that $\mathbf{1}^i \in \mathbb{R}^{|\mathcal{I}|}$ is a one-hot vector where the i^{th} value is 1, CE represents the Cross-Entropy function, and $\Theta(\{S_1^u, \ldots, S_t^u\}) \in \mathbb{R}^{|\mathcal{I}|}$ is the next-item prediction vector generated by the model for a user u, given their historical item sequence $\{S_1^u, \ldots, S_t^u\}$. Given the sequential data, we define an instance (X_n) as a pair consisting of an observed item sequence and the ground-truth next-item, i.e., $(\{S_1^u, \ldots, S_t^u\}, S_{t+1}^u)$.

Measuring Model Stability. Recent work [36] has shown that existing sequential recommenders generate unstable predictions when subjected to input data perturbations. Specifically, consider two scenarios. First, a recommender model Θ is trained on the original training data and generates recommendation lists $R_\Theta^{X_n}$ for all test instances X_n in the test data X_{test}. Second, another model Θ', which shares the same initial parameters as Θ, is trained on perturbed training data and produces rank lists $R_{\Theta'}^{X_n}$ for all $X_n \in X_{test}$. If the original model Θ is robust against input perturbation, then $R_\Theta^{X_n}$ and $R_{\Theta'}^{X_n}$ should be highly similar for all $X_n \in X_{test}$. However, existing work [36] has shown that even if there are minor changes to the training data, then $R_{\Theta'}^{X_n}$ is drastically different from $R_\Theta^{X_n}$ for all $X_n \in X_{test}$. To quantify the stability of the model Θ against input perturbations, we use the Rank List Stability (RLS) [36] metric:

$$RLS = \frac{1}{|X_{test}|} \sum_{\forall X_n \in X_{test}} similarity(R_\Theta^{X_n}, R_{\Theta'}^{X_n}), \quad (2)$$

where $similarity(A, B)$ denotes a similarity function between two rank lists A and B. Following [36], we use Rank-biased Overlap (RBO) [54] and Top-K Jaccard Similarity [21] as similarity functions.
(1) **Rank-biased Overlap (RBO)**: RBO [54] measures the similarity of orderings between two rank lists. A higher RBO indicates two rank lists are similar, and RBO values lie between 0 and 1. RBO gives higher importance to the similarity in the top part of the rank lists than the bottom part, making it our primary metric and preferable compared to other metrics like Kendall's Tau [25]. RBO of two rank lists A and B with $|\mathcal{I}|$ items is defined as follows.

$$RBO(A, B) = (1 - p) \sum_{d=1}^{|\mathcal{I}|} p^{d-1} \frac{|A[1:d] \cap B[1:d]|}{d},$$

where p is a hyperparameter (recommended value: 0.9).
(2) **Top-K Jaccard similarity**: Jaccard similarity ($Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$) is used to calculate the ratio of common top-K items between two rank lists, without considering the item ordering. The score ranges from 0 to 1, and a higher score indicates the top-K items in two rank lists are similar (indicating higher model stability). Regarding K, we use $K = 10$ as it is common practice [16, 27].

Scope: Interaction-level Perturbations. Similar to prior work [36], we assume **interaction-level minor perturbations**. Interaction perturbations are the smallest perturbation compared to user or item perturbations. Minor perturbations indicate that only a small number of interactions (e.g., 0.1%) can be perturbed in the training data. Naturally, larger perturbations will result in a greater decrease in stability. Possible perturbations include injecting interactions, deleting interactions, replacing items of interactions with other items, or a mix of them. To find such interactions to perturb, we can employ various perturbation algorithms for recommender systems [36, 40, 50, 55, 63, 65].

4 PROBLEM SETUP

Goal. Let Θ and Θ' be recommendation models trained with the original and perturbed training data, respectively. Also, let $R_\Theta^{X_n}$ and $R_{\Theta'}^{X_n}$ be the rank lists generated for a test instance X_n without and with perturbations, respectively. Then, our goal is to fine-tune Θ' with our proposed method FINEST, so that $R_{\Theta'}^{X_n}$ would be identical to $R_\Theta^{X_n}$ for all $X_n \in X_{test}$. Let $A[i]$ represent the i^{th} item in a rank list A, and $|\mathcal{I}|$ be the number of items. Then, formally, our objective is to ensure that: $R_{\Theta'}^{X_n}[i] = R_\Theta^{X_n}[i], \forall i$ from 1 to $|\mathcal{I}|$ for all $X_n \in X_{test}$ after fine-tuning with FINEST.

Assumptions. (1) The specific training interactions that are perturbed are not known during the fine-tuning. Thus, a fine-tuning model needs to be created that is robust regardless of the various interaction perturbations. (2) As model designers aiming to increase the model stability, we naturally have access to all the training data and the recommendation model (e.g., model parameters).

Measuring Training Effectiveness against Perturbations. We first measure the stability of a base recommender model Θ_{Base}, which is trained with the typical next-item prediction objective. The stability of Θ_{Base} is quantified by the RLS metrics using Equation (2), where high RLS values indicate the model is stable. Next, we fine-tune Θ_{Base} with FINEST (we call this fine-tuned model Θ_{FINEST}) and compute its stability using the RLS metrics. FINEST is successful if the stability of Θ_{FINEST} is higher than Θ_{Base}.

5 PROPOSED METHODOLOGY

We introduce a fine-tuning method called FINEST to enhance the stability of sequential recommender systems against input perturbations. FINEST simulates perturbation scenarios in the training data and aims to maximize the model's stability against such emulated perturbations with a rank-aware regularization function. Algorithm 1 and Fig. 2 summarize the main steps of FINEST. In **Step 1**, the base recommender Θ_{Base} is used to generate ranked item lists for all training instances $X_n \in X_{train}$: $R_{\Theta_{Base}}^{X_n}$. These lists will serve as reference rank lists for fine-tuning the model. Next, the base recommender Θ_{Base} is fine-tuned with FINEST for T epochs (steps 2–5), with T being a hyperparameter.

5.1 Step 2: Perturbation Simulations

Applying random perturbations on training data has contributed to enhancing model stability against input data perturbations in computer vision [13, 29, 41] and NLP [46]. Taking inspiration from this, FINEST simulates a pseudo-perturbation by randomly sampling training interactions (with a sampling ratio R) and perturbing them every epoch. Perturbations include one of three actions with equal probability: the interaction can be *deleted*, the interaction's item can be *replaced* with another, or a new interaction can be *inserted* before it. Re-sampling in every epoch ensures that the recommender model sees many variations of the input data and is

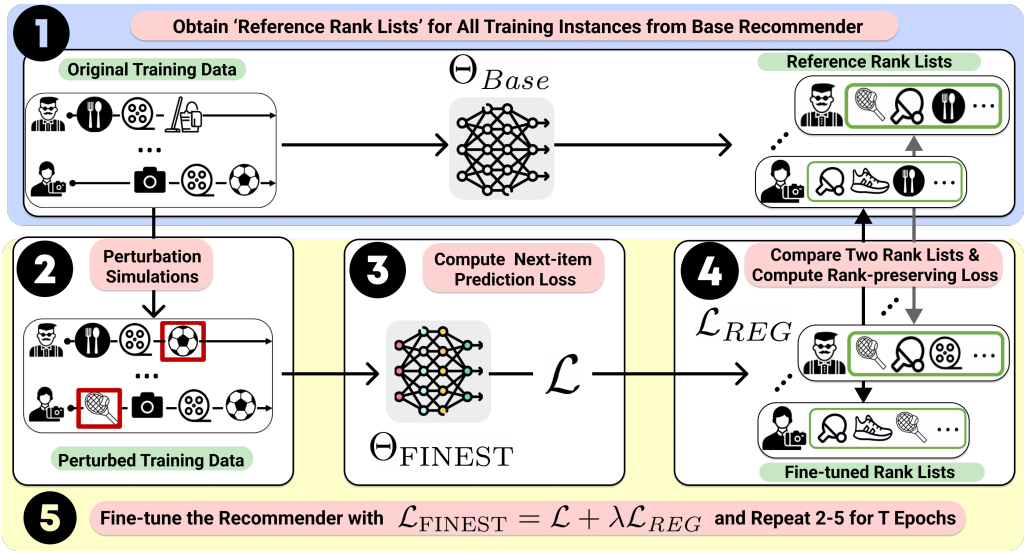


Figure 2: Overview of stabilizing a recommender model via fine-tuning with FINEST. First, we obtain reference recommendations for all training instances from a given recommendation model. Next, randomly sampled and perturbed data (changing every epoch) is to fine-tune the model. FINEST adds rank-preserving regularization to minimize differences between the reference and fine-tuned rank lists (generated under pseudo-perturbations). By simulating perturbations, FINEST can generate stable rank lists even in the presence of actual input perturbations.

Algorithm 1: FINEST: FINE-tuning for model STability

Input : A base recommender Θ_{Base}, training data X_{train}, sampling ratio R, number of sampled items K, number of fine-tuning epochs T, regularization constants $\lambda, \lambda_1, \lambda_2$.
Output: A fine-tuned recommender Θ_{FINEST}

▷ **`Step 1. Generate Reference Rank Lists`**

Generate reference rank lists $R^{X_n}_{\Theta_{Base}}, \forall X_n \in X_{train}$ using a given recommendation model Θ_{Base}
$\Theta_{\text{FINEST}} \longleftarrow \Theta_{Base}$
for *Fine-tuning epoch* $\in [1, \ldots, T]$ **do**
  ▷ **`Step 2. Training data is pseudo-perturbed`**
  Perform random sampling of interactions with the ratio R
  Perturb interactions by deletion, replacement, or insertion with equal probability; $X_{pert} \longleftarrow$ the set of perturbed interactions
  ▷ **`Step 3. Perform Next-item Prediction`**
  Calculate the loss $\mathcal{L}$ using Eq. (1) with perturbed training data
  ▷ **`Step 4. Rank-preserving Regularization`**
  Compute recommendation prediction scores of top-$2K$ items using Θ_{FINEST} for all training instances in $X_n \in X_{train} \backslash X_{pert}$
  Compute the regularization loss $\mathcal{L}_{REG}$ using Eq. (4)
  ▷ **`Step 5. Fine-tune to Stabilize Rank Lists`**
  Update the model parameters Θ_{FINEST} using Eq. (5)
Return the fine-tuned recommendation model Θ_{FINEST}

able to learn to make accurate and stable predictions regardless of a specific perturbation.

An example of perturbing training data by insertion is shown in Fig. 2. In an insertion perturbation of an interaction (u, i, t), we inject the least popular item into u's sequence with a timestamp right before t (e.g., $t-1$). Similarly, in an item replacement perturbation, the target item of the interaction is replaced with the least popular item in the dataset. In both cases, the least popular item is selected as it leads to the lowest RLS metrics of the recommender model compared to other items [36].

5.2 Step 3: Next-item Prediction on Perturbed Simulations

The perturbed data is used to fine-tune Θ_{Base}. Specifically, the parameters of Θ_{Base} are used to initialize Θ_{FINEST}. It is then fine-tuned for T epochs on the next-item prediction loss using the perturbed data as input (see Eq. (1)). This loss will be added with the rank-preserving regularization loss in the next step.

5.3 Step 4: Rank-preserving Regularization

The perturbed training data generated in Step 2 is used to fine-tune the recommender model Θ_{FINEST}. Θ_{FINEST} will generate rank lists $R^{X_n}_{\Theta_{\text{FINEST}}}$ for all training instances. However, due to the pseudo-perturbations, the generated rank lists $R^{X_n}_{\Theta_{\text{FINEST}}}$ can be different from the reference rank lists $R^{X_n}_{\Theta_{Base}}$ for all $X_n \in X_{train}$. The goal of the rank-preserving regularization is to ensure the two rank lists are identical after fine-tuning, meaning the rank lists do not change despite the perturbation.

Ideally, we need to preserve the ranking of all items in the reference rank lists to guarantee perfect model stability. However, keeping the ranks of all items in all the rank lists is computationally prohibitive (e.g., when there are millions of items in e-commerce). Therefore, we only aim to preserve the rank of the top-K items from each reference rank list. We focus on these top items because they are more likely to be viewed by end users. Formally, for each training instance $X_n \in X_{train}$, we perform top-$2K$ sampling on the reference rank list $R^{X_n}_{\Theta_{Base}}$ and represent it as $R^{X_n}_{\Theta_{Base}}[1:2K]$. Note that the other sampled items (from top-$K+1$ to top-$2K$) will be used for the rank-preserving loss computation later.

FINEST creates a novel rank-aware regularization loss $\mathcal{L}_{REG}(X_n)$ for any training instance $X_n \in X_{train}$. Let us assume the top-$2K$

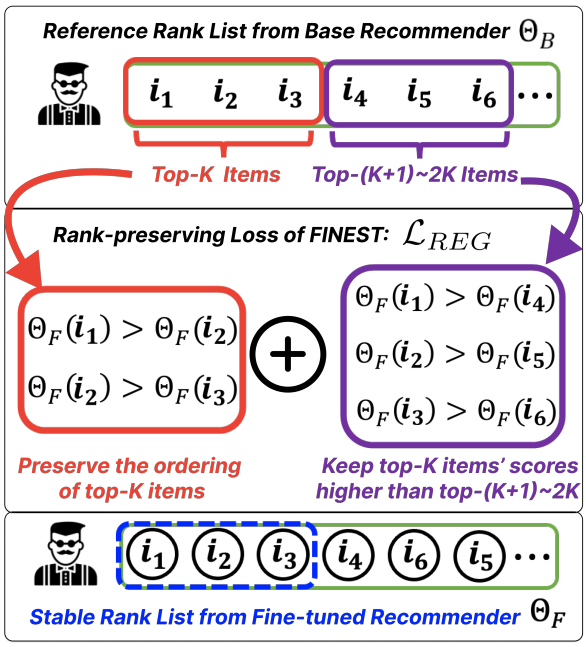


Figure 3: An illustration of the rank-preserving regularization (Eq. (3)) used in FINEST. The regularizer aims to preserve the ordering and high prediction scores of the top-K items in the reference rank lists. With the regularizer, the recommender can generate a similar top-K recommendation to the reference one under perturbations. Θ_B and Θ_F indicate Θ_{Base} and Θ_{FINEST}, respectively.

items in the reference rank list $R^{X_n}_{\Theta_{Base}}[1:2K]$ are $\{i_1, \ldots, i_{2K}\}$ and Θ_F is a simplified notation of Θ_{FINEST}. Then,

$$\mathcal{L}_{REG}(X_n) = \overbrace{\sum_{k=1}^{K-1} \max(\Theta_F(i_{k+1}) - \Theta_F(i_k) + \lambda_1, 0)}^{\text{Penalize if the relative ordering of top-}K\text{ items is violated}} + \overbrace{\sum_{k=1}^{K} \max(\Theta_F(i_{k+K}) - \Theta_F(i_k) + \lambda_2, 0)}^{\text{Penalize if higher prediction score to } K+1 \text{ to } 2K \text{ ranked items than top-}K}, \quad (3)$$

where λ_1 and λ_2 are margin values (hyperparameters). $k = 1$ is the first, i.e., indicating the highest-ranked item.

The first loss term penalizes if Θ_F gives a higher prediction score to i_{k+1} compared to i_k. Thus, the role of the first loss term is to ensure that the relative ordering of top-K items in $R^{X_n}_{\Theta_{Base}}$ is also maintained in $R^{X_n}_{\Theta_{\text{FINEST}}}$. The second loss term penalizes if Θ_F gives higher prediction scores to "competitive" items (i.e., $\{i_{K+1}, \ldots, i_{2K}\}$) than to the desired top-K items $\{i_1, \ldots, i_K\}$. Thus, the role of the second loss term is to place the top-K items from the reference rank list at the top part of the fine-tuned rank list. Both terms ensure that the relative positions and ordering of the top-K sampled items are the same in both rank lists. This is visualized in Fig. 3. We underscore that items $\{i_1, \ldots, i_{2K}\}$ **are obtained from the reference rank list** $R^{X_n}_{\Theta_{Base}}$, not the fine-tuned rank list $R^{X_n}_{\Theta_{\text{FINEST}}}$.

Table 2: Recommendation datasets used for experiments.

Name	Users	Items	Interactions	Descriptions
LastFM	980	1,000	1,293,103	Music playing history
Foursquare	2,106	5,597	192,602	Point-of-Interest check-in
Reddit	4,675	953	134,489	Subreddit posting history

The total regularization loss over all non-perturbed training instances is defined as follows:

$$\mathcal{L}_{REG} = \sum_{\forall X_n \in X_{train} \setminus X_{pert}} \mathcal{L}_{REG}(X_n), \quad (4)$$

where X_{pert} is the set of perturbed instances in the current epoch. $\mathcal{L}_{REG}$ is computed only for all non-perturbed instances $X_{train} \setminus X_{pert}$ of the current epoch. Perturbed instances are excluded because reference rank lists can be unavailable for perturbed instances.

It is important to *highlight the difference between the distillation loss [63] and the proposed rank-preserving loss.* Yue et al. [63] only choose randomly sampled negative items as "competitors" in the second loss term of Eq. (3). However, the random selection scheme does not help in preserving top-K ranking if the chosen negative samples are low-ranked in the reference rank lists.

5.4 Step 5: Total Loss

Overall, the total loss of FINEST simultaneously optimizes for the next-item prediction performance (Eq. (1)) and the rank-preserving regularization performance (Eq. (4)) as follows:

$$\mathcal{L}_{\text{FINEST}} = \mathcal{L} + \lambda \mathcal{L}_{REG}, \quad (5)$$

where λ indicates the regularization strength (a hyperparameter). It is essential to optimize both objectives together. If the next-item prediction loss $\mathcal{L}$ is not included, then the model may sacrifice next-item prediction performance in favor of the stability objective. This is undesirable as it will reduce the utility of the resulting model.

6 EXPERIMENTS

In this section, we conduct rigorous experimental evaluations of our fine-tuning method, FINEST, to assess its effectiveness.

6.1 Experimental Settings

Datasets. We use three datasets that are widely used in the existing literature and span various domains. The statistics are listed in Table 2. Users with fewer than 10 interactions were filtered out.

- LastFM [15, 19, 22, 28]: This dataset consists of the music-playing history of users, represented as (user, music, timestamp).
- Foursquare [57, 58, 60, 61]: This dataset represents point-of-interest information, including user, location, and timestamp.
- Reddit [1, 27, 32, 37]: This dataset contains the posting history of users on subreddits, represented as (user, subreddit, timestamp).

Target Recommender Models. We aim to improve the model stability of the following state-of-the-art sequential recommenders.

- **TiSASRec [31]:** a self-attention based model that utilizes temporal features and positional embeddings for next-item prediction.
- **BERT4Rec [45]:** a bidirectional Transformer-based model that uses masked language modeling for sequential recommendations.
- **LSTM [20]:** a Long Short-Term Memory (LSTM)-based model that can learn long-term dependencies using LSTM architecture.

Table 3: Effectiveness of various fine-tuning methods for recommendation models on top-2 largest datasets. FINEST is the best fine-tuning method as per enhancing model stability (measured by RLS metrics) against random and CASPER [36] deletion perturbations, with statistical significance (p-values < 0.05) in all cases. FINEST performs the best for other types of perturbations and datasets as well. F.T. = Fine-tuning.

a LastFM Dataset (Music Recommendation; 1.3 Million Interactions)

Input Perturbations	Random Deletion Perturbations						CASPER [36] Deletion Perturbations					
Recommender Models	TiSASRec [31]		BERT4Rec [45]		LSTM [20]		TiSASRec [31]		BERT4Rec [45]		LSTM [20]	
RLS Metrics	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard
Original (No F.T.)	0.7528	0.2750	0.7535	0.3164	0.7691	0.2693	0.6938	0.1999	0.7538	0.3157	0.7002	0.1715
Random F.T.	0.7621	0.2953	0.7762	0.3728	0.7865	0.3002	0.7017	0.2153	0.7734	0.3662	0.6989	0.1665
Earliest-Random F.T.	0.7600	0.2902	0.7756	0.3637	0.7804	0.2924	0.7016	0.2124	0.7735	0.3640	0.7000	0.1676
Latest-Random F.T.	0.7630	0.3015	0.7844	0.3797	0.7744	0.2802	0.7034	0.2200	0.7773	0.3672	0.6986	0.1663
APT [56] F.T.	0.7636	0.2967	0.7770	0.3681	0.7792	0.2901	0.7012	0.2119	0.7746	0.3634	0.6992	0.1676
ACAE [59] F.T.	0.7634	0.2944	0.7701	0.3519	0.7792	0.2861	0.6992	0.2103	0.7725	0.3651	0.7000	0.1689
FINEST (Proposed)	**0.9207**	**0.6594**	**0.8353**	**0.4824**	**0.9038**	**0.5903**	**0.8727**	**0.5193**	**0.8324**	**0.4757**	**0.7872**	**0.3351**
Rel. % Improvements	**+20.6 %**	**+119%**	**+6.5%**	**+27.0%**	**+14.9%**	**+96.6%**	**+24.1%**	**+136%**	**+7.5%**	**+29.5%**	**+12.4%**	**+95.4%**

b Foursquare Dataset (POI Recommendation; 0.2 Million Interactions)

Input Perturbations	Random Deletion Perturbations						CASPER [36] Deletion Perturbations					
Recommender Models	TiSASRec [31]		BERT4Rec [45]		LSTM [20]		TiSASRec [31]		BERT4Rec [45]		LSTM [20]	
RLS Metrics	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard	RBO	Jaccard
Original (No F.T.)	0.7680	0.2725	0.7948	0.3542	0.7096	0.1682	0.7787	0.2840	0.7964	0.3566	0.6459	0.1137
Random F.T.	0.7628	0.2618	0.8190	0.4277	0.7078	0.1713	0.7694	0.2660	0.8149	0.4151	0.6470	0.1181
Earliest-Random F.T.	0.7643	0.2571	0.8112	0.4042	0.7152	0.1772	0.7738	0.2663	0.8149	0.4141	0.6476	0.1182
Latest-Random F.T.	0.7582	0.2547	0.8150	0.4151	0.7127	0.1748	0.7622	0.2626	0.8161	0.4173	0.6463	0.1157
APT [56] F.T.	0.7618	0.2965	0.8081	0.3919	0.7077	0.1685	0.7904	0.3247	0.8088	0.4001	0.6482	0.1173
ACAE [59] F.T.	0.7798	0.2918	0.8045	0.3834	0.7137	0.1689	0.7866	0.3025	0.8068	0.3879	0.6503	0.1188
FINEST (Proposed)	**0.9368**	**0.6510**	**0.8824**	**0.5080**	**0.8452**	**0.4116**	**0.9374**	**0.6501**	**0.8794**	**0.5056**	**0.7358**	**0.2168**
Rel. % Improvements	**+20.1 %**	**+120%**	**+7.7%**	**+18.8%**	**+18.2%**	**+132%**	**+18.6%**	**+100%**	**+7.8%**	**+21.2%**	**+13.1%**	**+82.5%**

Next-item Prediction Metrics. To evaluate the next-item prediction accuracy, we use two popular metrics, namely, Mean Reciprocal Rank (MRR) and Recall@K (typically $K = 10$ [16, 27]). The metric values are between 0 to 1, and higher values are better. These metrics only focus on the rank of the ground-truth next item, not the ordering of all items in a rank list. Thus, they are *unsuitable* to measure the model stability against input perturbations.

Training Data Perturbation Methods.

• **Random, Earliest-Random, and Latest-Random:** Random perturbation manipulates randomly chosen interactions among all training interactions, while Earliest-Random and Latest-Random approaches perturb randomly selected interactions among the first and last 10% interactions of users, respectively.

• **Data-free [63]:** Data-free [63] perturbation generates fake user profiles via T-FGSM [14] to promote target items. To adapt it to our setting, (1) we find the most similar users in the original training data to those fake users, and (2) either inject generated fake interactions to such similar users' sequences or perform deletions or item replacements to similar users' interactions.

• **CASPER [36]:** CASPER [36] is the *state-of-the-art* interaction-level perturbation for sequential recommendation models. FINEST employs a graph-based approximation to find the most effective perturbation in the training data to alter RLS metrics.

Note that we use the least popular item for injection and item replacement perturbations since they are the most effective [36]. FINEST **does not know** what perturbations will be applied to the training data during its fine-tuning process.

Baseline Fine-tuning Methods to Compare against FINEST.

• **Original:** It trains a recommender model on original training data (without fine-tuning) with standard next-item prediction loss.

• **Random, Earliest-Random, and Latest-Random:** The Random method perturbs 1% of random training interactions for every epoch (either deletion, insertion, or replacement) and fine-tunes a recommender model with the perturbed data. The Earliest-Random and Latest-Random randomly perturb 1% of interactions in the first and last 10% (based on timestamps) of the training data and fine-tune the model with the perturbed data, respectively.

• **Adversarial Poisoning Training (APT) [56]:** APT is the **state-of-the-art** adversarial training method that fine-tunes a recommendation model using perturbed training data, including fake user profiles. Since it only works for matrix factorization-based models, we replace the fake user generation part with the Data-free [63] model and fine-tune the recommender with the perturbed data.

• **ACAE [59]:** ACAE is another **state-of-the-art** adversarial training model that adds gradient-based noise (found by the fast gradient method [14]) to the model parameters while fine-tuning a recommendation model. To adapt it to our setting, we add noise to the input sequence embeddings instead of model parameters.

We exclude other fine-tuning methods designed for multimodal recommender systems [3, 48] which require additional input data like images or show similar or worse performance [2, 11, 18, 55, 65] than existing baselines such as APT [56] or ACAE [59].

Experimental Setup. We use the first 90% of interactions of each user for training and validation, and the rest of the interactions are used for testing. To train recommender models, we use the hyperparameters suggested in their original publications. Other common hyperparameters are set as follows: the maximum training

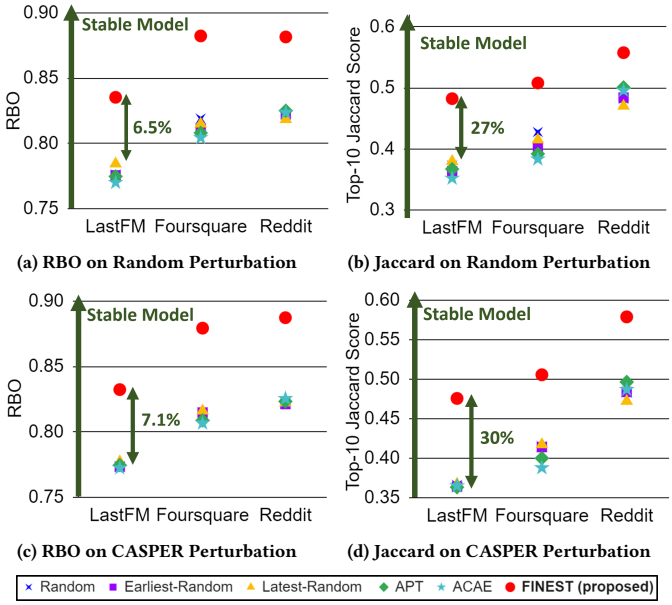
Table 4: *Next-item prediction performance of FINEST on LastFM and Foursquare datasets (no perturbations).* FINEST successfully preserves or enhances next-item metrics of all recommendation models. Results with * indicate statistical significance (p-value < 0.05).

a LastFM Dataset

Recommender	TiSASRec [31]		BERT4Rec [45]		LSTM [20]	
Next-item Metrics	MRR	Recall @10	MRR	Recall @10	MRR	Recall @10
Original Model	0.154	0.277	0.158	0.318	0.156	0.235
FINEST (Proposed)	**0.164***	**0.299***	**0.168***	**0.343***	**0.169***	**0.252***

b Foursquare Dataset

Recommender	TiSASRec [31]		BERT4Rec [45]		LSTM [20]	
Next-item Metrics	MRR	Recall @10	MRR	Recall @10	MRR	Recall @10
Original Model	0.082	0.137	0.162	0.267	0.096	0.166
FINEST (Proposed)	**0.088**	**0.143**	**0.175***	**0.283***	**0.102**	**0.174**



(a) RBO on Random Perturbation **(b) Jaccard on Random Perturbation**

(c) RBO on CASPER Perturbation **(d) Jaccard on CASPER Perturbation**

Figure 4: *Stability of the BERT4Rec model fine-tuned with diverse methods against random and CASPER [36] deletion perturbations across different datasets.* FINEST generates the most stable model against both perturbations as per RBO and Top-10 Jaccard Similarity.

epoch is set to 100, a learning rate is set to 0.001, and the embedding dimension is set to 128. For all recommendation models, the maximum sequence length per user is set to 50. We also perturb 0.1% of training interactions, as described in Section 3. We repeat all experiments three times and report average values of RLS and next-item metrics. For FINEST, we use the following hyperparameters (found by grid searches on validation data). The sampling ratio of interactions for perturbation simulations is set to 1%, and we sample the top-200 items for regularization, and the regularization coefficients $\lambda, \lambda_1, \lambda_2$ are set to 1.0, 0.1, and 0.1, respectively. We assign 50 epochs for fine-tuning. To measure statistical significance, we use the one-tailed t-test.

6.2 Effectiveness of FINEST

Fine-tuning method comparison on the LastFM and Foursquare datasets. In Table 3, we compare the performance of all fine-tuning methods on all three recommendation models against random and CASPER [36] deletion perturbations on the LastFM and Foursquare datasets. We highlight the results on these two datasets as they are the top-2 largest ones as per the number of interactions. Each column shows the original method without any fine-tuning and the best fine-tuning method with the highest RLS value. Note that most stability analysis results against the Data-free [63] perturbation are omitted due to the space limits (some of them are summarized in Table 5), but they show a similar trend to Table 3.

Our proposed fine-tuning method FINEST **outperforms** all of the baselines across all recommender systems, with statistical significance in all cases (p-values < 0.05). FINEST demonstrates significant improvements in RLS metrics compared to the results of original training and baselines. For instance, on the LSTM model (the most susceptible one against CASPER perturbation), FINEST shows at least 12.4% RBO and 82.5% Jaccard improvements versus the best baseline. Even on the BERT4Rec model (the most stable one against CASPER perturbation), FINEST still exhibits at least 7.5% RBO and 21.2% Jaccard score boosts versus the best baseline. Baseline fine-tuning methods have limitations in improving model stability since they do not incorporate the rank list preservation component in their fine-tuning. We observe the same trend for other types of perturbations and datasets (see Fig. 4 and Table 5). Thus, with FINEST, the state-of-the-art recommender systems can generate stable and trustworthy rank lists of items even after perturbations.

Impact of FINEST on next-item prediction accuracy. Fine-tuning the recommender with randomly sampled perturbations can increase or preserve the next-item prediction accuracy (e.g., MRR and Recall@10). *This is due to the implicit data augmentation and cleaning effect from the perturbed training examples.* We validate the improvements of both next-item prediction performance and model stability in Table 4 (on the LastFM and Foursquare datasets, with no perturbations). Table 4 demonstrates that FINEST can boost the model stability without sacrificing its next-item prediction accuracy with statistical significance in most cases. TiSASRec shows relatively lower next-item prediction performance as it is optimized for sampled metrics, which computes ranking with negative items during the test. For more details, please refer to the paper [26]. Meanwhile, the other models are optimized over all items.

Fine-tuning method comparison on different datasets. We further evaluate the effectiveness of FINEST versus the baselines on the BERT4Rec model (which shows high accuracy and stability) across various datasets. The results are shown in Fig. 4. We find that FINEST exhibits the highest model stability with statistical significance (p-values < 0.05) across all datasets and two perturbations, as per both RLS metrics. For instance, on the LastFM dataset (the largest one in terms of the number of interactions), FINEST offers at least 6.5% stability improvements compared to baselines in terms of RBO and at least 27% in top-10 Jaccard similarity.

Effectiveness of FINEST against diverse perturbation types. We test the generalizability of FINEST in boosting model stability against various interaction-level perturbation algorithms and types. As shown in Table 5, FINEST improves the RBO metric of the BERT4Rec model against various perturbations on the LastFM dataset, across three perturbation types. The Jaccard metric shows a similar trend. Against CASPER [36], FINEST boosts the RBO of the

Table 5: *RLS metric (RBO) of FINEST against diverse interaction-level perturbation algorithms and types on BERT4Rec model and LastFM dataset.* FINEST successfully enhances the model stability of BERT4Rec against all types of input perturbations with statistical significance. Pert. = Perturbations, R. = Random, E.R. = Earliest-Random, L.R. = Latest-Random, D.F. = Data-free [63], CAS. = CASPER [36], Orig. = Original Model.

a Perturbation Type: Insertion

Pert.	R.	E.R.	L.R.	D.F.	CAS.
Orig.	0.75	0.75	0.76	0.76	0.75
FINEST	**0.83**	**0.84**	**0.84**	**0.84**	**0.84**

b Perturbation Type: Item Replacement

Pert.	R.	E.R.	L.R.	D.F.	CAS.
Orig.	0.91	0.96	0.88	0.96	0.84
FINEST	**0.94**	**0.97**	**0.91**	**0.97**	**0.88**

c Perturbation Type: Deletion

Pert.	R.	E.R.	L.R.	D.F.	CAS.
Orig.	0.75	0.75	0.75	0.75	0.75
FINEST	**0.84**	**0.83**	**0.84**	**0.84**	**0.83**

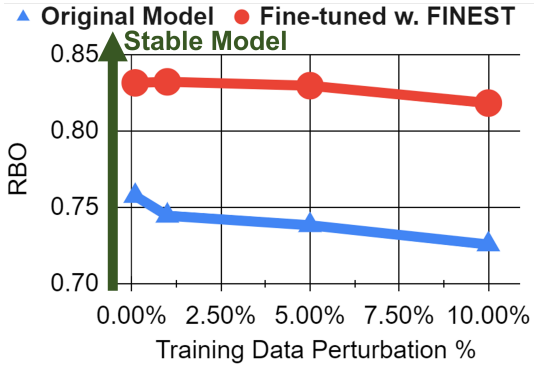


Figure 5: *Stability of BERT4Rec fine-tuned with and without FINEST as per the number of input perturbations on the LastFM dataset.*

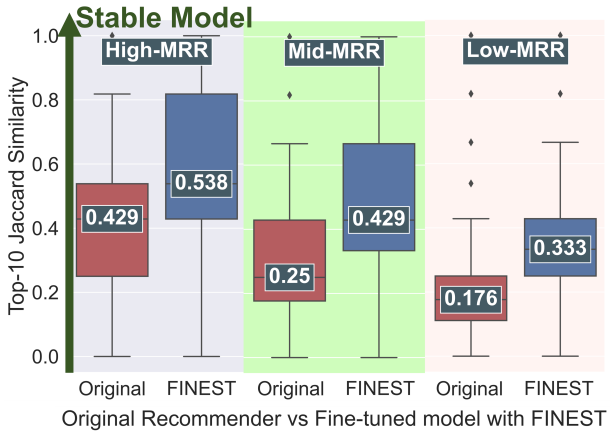


Figure 6: FINEST enhances the stability of recommendations across all user groups with different next-item prediction accuracies. These results are for recommendations from the BERT4Rec model on the LastFM dataset with respect to the CASPER [36] perturbation.

model by at least 5.4%. These results indicate that FINEST consistently enhances model stability, regardless of input perturbations. It is worth noting that insertion and deletion perturbations lead to similar model stability, while item replacement perturbations are the least effective. This is because replacing items has little impact on constructing training sequences of items than injecting or deletion interactions. FINEST can also enhance the model stability against a mix of the above interaction perturbations (omitted due to the space limits).

6.3 Model Stability against Large Perturbations

We test how much the RLS metric of FINEST changes with respect to the number of input perturbations, since more perturbations will naturally lower the model stability further. Fig. 5 shows the RBO scores of the BERT4Rec model trained with and without FINEST on the LastFM dataset against CASPER deletion perturbations while varying the perturbation scale from 0.1% to 10%. FINEST provides significant improvements in the model stability compared to the original model across all perturbation scales.

6.4 Scalability of FINEST

The time and space complexities of FINEST scale near-linearly to the number of interactions and items in a dataset. Empirically, we also confirm that FINEST can enhance the stability of recommenders on large-scale datasets such as MovieLens-10M [17] (72K users, 10K items, and 10M interactions) or Steam [24] (2.6M users, 15K items, and 7.8M interactions) datasets. For instance, BERT4Rec with FINEST shows a 12% improvement in the RBO metric (0.763 → 0.853) compared to the original BERT4Rec on the Steam dataset with random perturbations.

Table 6: Ablation study of the key components of FINEST.

	RLS metrics		Next-item metrics	
Fine-tuning Methods / Metrics	**RBO**	**Jaccard**	**MRR**	**Recall**
Original (no fine-tune)	0.7538	0.3157	0.1580	0.3181
FINEST without Perturbation Simulation	0.8124	0.4349	0.1634	0.3324
FINEST without Top-K Regularization	0.7734	0.3662	0.1665	0.3366
FINEST without First Regularization Term	0.8268	0.4658	0.1663	0.3379
FINEST without Second Regularization Term	0.7955	0.4245	**0.1713**	**0.3513**
FINEST (proposed)	**0.8324**	**0.4757**	0.1682	0.3434

6.5 Effectiveness on Different User Groups

It has been shown that input perturbations can disproportionately affect users' recommendation results [36]. A stability analysis result of the BERT4Rec model on the LastFM dataset against the CASPER perturbation supports that observation. As shown in Fig. 6, a *low-accuracy* user group (with the lowest 20% MRR metric on average among all users) receives unstable recommendations compared to the *high-accuracy* group. This can raise fairness concerns between user groups similar to "the rich get richer" problem. FINEST can mitigate this issue by enhancing the stability of the model, thereby narrowing the relative stability difference between the two groups (e.g., 143% without FINEST → 62% with FINEST).

6.6 Ablation Studies of FINEST

We verify the contributions of the perturbation simulation and rank-preserving regularization of FINEST by measuring the model stability after removing each component. Table 6 shows the ablation study results on the BERT4Rec model and LastFM dataset against CASPER deletion perturbations in terms of RLS and next-item

metrics. We observe that all variants of FINEST outperform the original training (without fine-tuning) in all metrics. Among the variants, we see that the model without the perturbation simulation performs better than the model without the regularization, implying that the top-K regularization has a higher impact on enhancing the model stability. Regarding the regularization function, the "score-preserving" component (second term in Eq. (3)) is more effective in terms of RLS metrics than the "ordering-preserving" component (first term in Eq. (3)). In summary, having both components of FINEST together results in the highest model stability.

6.7 Hyperparameter Sensitivity of FINEST

Figure 7 exhibits the hyperparameter sensitivity of FINEST with respect to RLS and next-item metrics on the BERT4Rec model and LastFM dataset against CASPER deletion perturbations. We change one hyperparameter while fixing all the others to the default values stated in Section 6.1. We find that both metrics improve as fine-tuning continues, and the improvements saturate after sufficient epochs (e.g., 50) of fine-tuning are done. Regarding the sampling ratio, we observe the trade-off between RLS and next-item metrics as the ratio increases. In practice, a small value (e.g., 1%) is preferred as a high value can hurt the next-item metrics. A medium number of top-K items (e.g., 100) is best for FINEST since a small value can have a minor impact on preserving the rank lists, and a large value can reduce the scalability of FINEST. Finally, a medium value of λ (e.g., 1) leads to high RLS and next-item metrics, as a small value limits the effect of the ranking-preserving regularization, while a large value can lead to inaccurate next-item predictions.

7 DISCUSSION

Why is Fine-tuning Selected over Retraining? One may wonder whether training with FINEST from scratch (instead of fine-tuning) is sufficient for achieving high model stability. There are two key reasons why fine-tuning is preferred. First, existing literature in recommender systems [18, 56, 59] has demonstrated that fine-tuning mechanisms should be applied when the given model starts to overfit [18] for the best model robustness, not when it is still in the early stage. Thus, it is better to apply the fine-tuning method FINEST after the model has been trained sufficiently, not from the beginning. Second, our fine-tuning process requires the rank lists of all training instances as a reference for the regularization. If the model is not fully trained as per next-item prediction accuracy, the reference lists will not be optimal. A pre-trained recommendation model ensures that appropriate reference lists are used. Third, fine-tuning techniques can be applied to existing pre-trained recommendation models, rather than requiring models to be trained from scratch. This makes fine-tuning techniques applicable even to deployed models that are typically trained extensively.

Should All Rank Lists be Stabilized with FINEST? FINEST fine-tunes a recommender to generate stable rank lists for all training instances. However, in some cases, the rank list is expected to change against perturbations. For instance, let us assume a cold-start user with very few interactions. If we perturb this user's interaction, the recommendation should be altered, since every single interaction of the cold-start user is crucial for its recommendation. Finding more types of rank lists not to stabilize is worth studying.

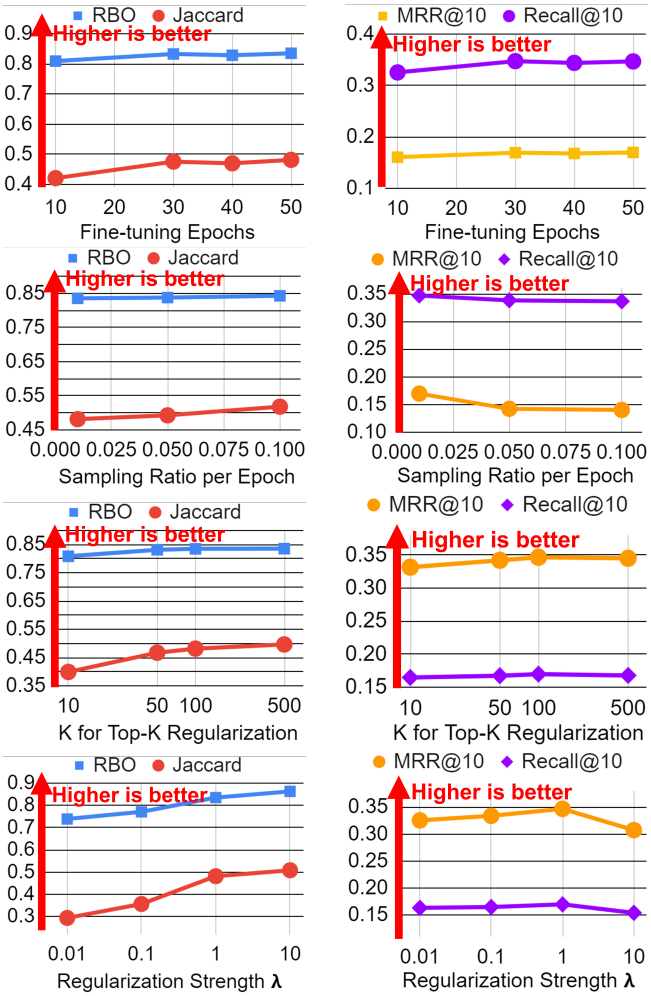


Figure 7: ***Hyperparameter sensitivity of FINEST on the BERT4Rec model and LastFM dataset against CASPER deletion perturbations.***

Handling Diverse Perturbation Methods. In this paper, we focused on enhancing model stability against interaction-level perturbations such as injection, deletion, item replacement, and a mix of them. However, in the real world, there can be various types of perturbations such as user-, item-, or embedding-level perturbations. While FINEST can be easily extended to user- and item-level perturbations by performing the perturbation simulation at the user or item level, extending FINEST to embedding-level perturbations is worth investigating as finding embedding perturbations for our simulations is non-trivial.

Non-Sequential Recommender Systems. As FINEST is optimized for sequential recommenders, its fine-tuning process should be modified for non-sequential recommendation models, such as collaborative filtering (CF). For instance, we can apply our rank-preserving regularization to each user instead of each training instance for CF-based recommenders. FINEST can be generalized to multimodal recommendation setup, where recommenders employ additional modalities such as text or image features for training and predictions. We leave the empirical validation of FINEST on such non-sequential recommenders as future work.

8 CONCLUSION

Our work paves the path toward robust and reliable recommendation systems by proposing a novel fine-tuning method with perturbation simulations and rank-preserving regularization. Future work includes extending FINEST to diverse recommendation models (e.g., reinforcement learning-based), other perturbation settings (e.g., embedding-level), and creating fine-tuning mechanisms for various content-aware recommendation models.

REFERENCES

[1] 2020. Reddit data dump. http://files.pushshift.io/reddit/.

[2] Vito Walter Anelli, Alejandro Bellogín, Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. MSAP: Multi-Step Adversarial Perturbations on Recommender Systems Embeddings. *FLAIRS* 34 (Apr. 2021).

[3] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. 2021. A study of defensive methods to protect visual recommendation against adversarial manipulation of images. In *SIGIR, ACM*.

[4] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A Formal Analysis of Recommendation Quality of Adversarially-trained Recommenders. In *CIKM*.

[5] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*. 274–283.

[6] Yuanjiang Cao, Xiaocong Chen, Lina Yao, Xianzhi Wang, and Wei Emma Zhang. 2020. Adversarial Attacks and Detection on Reinforcement Learning-Based Interactive Recommender Systems. In *SIGIR*.

[7] Pablo Castells, Neil Hurley, and Saul Vargas. 2022. Novelty and diversity in recommender systems. In *Recommender systems handbook*. 603–646.

[8] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial Attacks on an Oblivious Recommender. In *RecSys*.

[9] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation. In *RecSys*.

[10] T. Di Noia, D. Malitesta, and F. A. Merra. 2020. TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems. In *DSN-W*.

[11] Yali Du, Meng Fang, Jinfeng Yi, Chang Xu, Jun Cheng, and Dacheng Tao. 2018. Enhancing the robustness of neural collaborative filtering systems under malicious attacks. *IEEE Transactions on Multimedia* 21, 3 (2018).

[12] Michael D Ekstrand, Anubrata Das, Robin Burke, and Fernando Diaz. 2022. Fairness in recommender systems. In *Recommender systems handbook*. 679–707.

[13] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. 2021. Maxup: Lightweight adversarial training with data augmentation improves neural network training. In *CVPR*. 2474–2483.

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

[15] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *SIGKDD*.

[16] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and sequential user embeddings for large-scale music recommendation. In *RecSys*.

[17] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, Article 19 (Dec. 2015), 19 pages.

[18] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*.

[19] Balázs Hidasi and Domonkos Tikk. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *ECML PKDD*.

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997).

[21] Paul Jaccard. 1912. The distribution of the flora in the alpine zone. 1. *New phytologist* 11, 2 (1912), 37–50.

[22] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2019. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *WSDM*.

[23] Dietmar Jannach and Michael Jugovac. 2019. Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems (TMIS)* 10, 4 (2019), 1–23.

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

[25] Maurice George Kendall. 1948. Rank correlation methods. (1948).

[26] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.

[27] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *SIGKDD*.

[28] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. In *SIGKDD*.

[29] Alexander Levine and Soheil Feizi. 2020. Robustness certificates for sparse adversarial attacks by randomized ablation. In *AAAI*, Vol. 34. 4585–4593.

[30] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[31] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*.

[32] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic graph collaborative filtering. In *ICDM*.

[33] Fang Liu and Ness Shroff. 2019. Data poisoning attacks on stochastic bandits. In *ICML*.

[34] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 4487–4496. https://www.aclweb.org/anthology/P19-1441

[35] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *EMNLP*.

[36] Sejoon Oh, Berk Ustun, Julian McAuley, and Srijan Kumar. 2022. Rank List Sensitivity of Recommender Systems to Interaction Perturbations. In *CIKM*.

[37] Shalini Pandey, George Karypis, and Jaideep Srivasatava. 2021. IACN: Influence-Aware and Attention-Based Co-evolutionary Network for Recommendation. In *PAKDD*.

[38] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *WWW*.

[39] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *RecSys*.

[40] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. 2020. Estimating Training Data Influence by Tracking Gradient Descent. *NeurIPS*.

[41] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *ICML*.

[42] João Sá, Vanessa Queiroz Marinho, Ana Rita Magalhães, Tiago Lacerda, and Diogo Goncalves. 2022. Diversity Vs Relevance: A Practical Multi-objective Study in Luxury Fashion Recommendations. In *SIGIR*. 2405–2409.

[43] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *ICLR*.

[44] J. Song, Z. Li, Z. Hu, Y. Wu, Z. Li, J. Li, and J. Gao. 2020. PoisonRec: An Adaptive Data Poisoning Framework for Attacking Black-box Recommender Systems. In *ICDE*.

[45] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*.

[46] Abigail Swenor. 2022. Using Random Perturbations to Mitigate Adversarial Attacks on NLP Models. *AAAI* (2022), 13142–13143.

[47] Huiyi Tan, Junfei Guo, and Yong Li. 2008. E-learning recommendation system. In *CSSE*, Vol. 5. IEEE, 430–433.

[48] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. 2019. Adversarial training towards robust multimedia recommender system. *TKDE* 32, 5 (2019), 855–867.

[49] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2289–2298.

[50] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *RecSys*.

[51] Thi Ngoc Trang Tran, Alexander Felfernig, Christoph Trattner, and Andreas Holzinger. 2021. Recommender systems in the healthcare domain: state-of-the-art and research issues. *Journal of Intelligent Information Systems* 57, 1 (2021).

[52] Jingkang Wang, Tianyun Zhang, Sijia Liu, Pin-Yu Chen, Jiacen Xu, Makan Fardad, and Bo Li. 2021. Adversarial Attack Generation Empowered by Min-Max Optimization. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[53] Yifan Wang, Weizhi Ma, Min Zhang*, Yiqun Liu, and Shaoping Ma. 2022. A Survey on the Fairness of Recommender Systems. *ACM Journal of the ACM (JACM)* (2022).

[54] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM TOIS* (2010).

[55] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2021. Triple Adversarial Learning for Influence based Poisoning Attack in Recommender Systems. In *SIGKDD*.

[56] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, Enhong Chen, and Senchao Yuan. 2021. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In *SIGIR*. 1074–1083.

[57] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *SIGKDD*.
[58] Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *SIGSPATIAL*. 458–461.
[59] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial collaborative neural network for robust recommendation. In *SIGIR*.
[60] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *SIGIR*.
[61] Quan Yuan, Gao Cong, and Aixin Sun. 2014. Graph-based point-of-interest recommendation with geographical and temporal influences. In *CIKM*.
[62] Xiaofang Yuan, Ji-Hyun Lee, Sun-Joong Kim, and Yoon-Hyun Kim. 2013. Toward a user-oriented recommendation system for real estate websites. *Information Systems* 38, 2 (2013), 231–243.
[63] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *RecSys*.
[64] Hengtong Zhang, Y. Li, B. Ding, and Jing Gao. 2020. Practical Data Poisoning Attack against Next-Item Recommendation. In *TheWebConf*.
[65] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and Jing Gao. 2021. Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data. In *SIGKDD*.
[66] Dávid Zibriczky12. 2016. Recommender systems meet finance: a literature review. In *Proc. 2nd Int. Workshop Personalization Recommender Syst.* 1–10.