Rank List Sensitivity of Recommender Systems to Interaction Perturbations

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

Prediction models can exhibit sensitivity with respect to training data: small changes in the training data can produce models that assign conflicting predictions to individual data points during test time. In this work, we study this sensitivity in recommender systems, where users' recommendations are drastically altered by minor perturbations in other unrelated users' interactions. We introduce a measure of stability for recommender systems, called *Rank List Sensitivity* (RLS), which measures how rank lists generated by a given recommender system at test time change as a result of a perturbation in the training data. We develop a method, CASPER, which uses cascading effect to identify the minimal and systematical perturbation to induce higher instability in a recommender system. Experiments on four datasets show that recommender models are overly sensitive to minor perturbations introduced randomly or via CASPER — even perturbing one random interaction of one user drastically changes the recommendation lists of all users. Importantly, with CASPER perturbation, the models generate more unstable recommendations for low-accuracy users (i.e., those who receive low-quality recommendations) than high-accuracy ones.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems, Model Stability, Input Data Perturbation

ACM Reference Format:
Sejoon Oh, Berk Ustun, Julian McAuley, and Srijan Kumar. 2022. Rank List Sensitivity of Recommender Systems to Interaction Perturbations. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557425

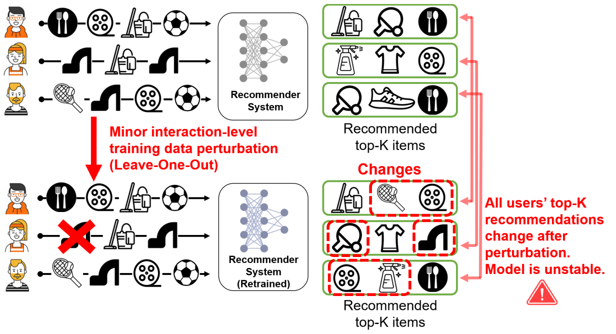

Recommender System
Minor interaction-level training data perturbation (Leave-One-Out)
Recommended top-K items
Changes
Recommender System (Retrained)
All users' top-K recommendations change after perturbation. Model is unstable.
Recommended top-K items

Figure 1: Small changes (e.g., leave-one-out perturbation) in the training data can produce recommender systems that output drastically different recommendations for *all* individual users.

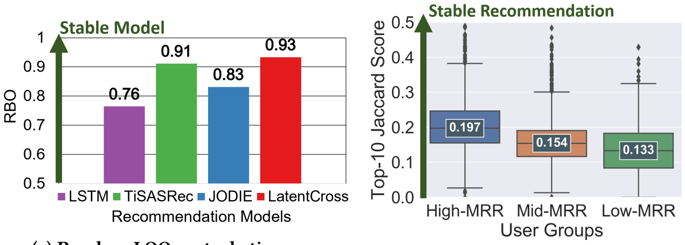
1 INTRODUCTION

Small changes in training data can produce large changes in outputs of machine learning models [7, 40, 47, 57, 59, 75]. Marx et al. [45] showed how classification tasks can often admit competing models that perform almost equally well in terms of an aggregate performance metric (e.g., error rate, AUC) but that assign conflicting predictions to individual data points. Likewise, Black and Fredrikson [6] showed how removing a point from a training dataset can produce models that assign drastically different predictions, and highlighted how this lack of stability disproportionately affects points with low confidence predictions.

The sensitivity with respect to minor data perturbations is especially meaningful and concerning in modern recommender systems – where data points pertain to user interactions. In this setting, sensitivity would imply that the recommendations for a user change due to small arbitrary changes in the training data from another unrelated user. This effect can be disruptive or even dangerous when recommendation systems are used for applications in healthcare, finance, education, and housing [62, 64, 79, 85]. Consider a system that recommends a specific treatment to a patient based on data from their electronic health record [64, 70]. In this setting, sensitivity would imply that the treatment recommendations for a patient by a given system could change due to noisy training data for another patient – e.g., due to errors introduced when digitizing handwritten notes or transcribing voice memos [13, 33, 61]. More broadly, this sensitivity could be introduced due to intentional manipulations – as a malicious adversary could inject noise into the training data to degrade the overall recommendation quality by producing

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557425



(a) Random LOO perturbation

(b) Stability across user groups

Figure 2: (a) Four recommendation models are shown to be unstable against minor perturbations — random leave-one-out perturbation in training data changes output rank lists of *all users* drastically. (b) Our proposed perturbation method, CASPER, lowers the stability, measured via Jaccard@10, of users with low accuracy the most.

low-quality recommendations for all users or even disproportionate damage on specific user or item groups [16, 18, 19, 58, 81].

These effects broadly underscore the need to *measure* the sensitivity in the development of recommender systems – so that model developers and end users can decide whether to use a specific recommendation, or whether to use recommender systems at all.

In this work, we study the sensitivity of recommender systems, so that practitioners can measure the stability of their models and make informed decisions in model development and deployment. Our problem statement is: ***Can an arbitrary change in a single data point in the training data change the recommendations for other data points? If so, what is the maximum change in recommendations possible with that change?***

We propose a novel framework to measure the stability by comparing two recommendation lists for each test interaction – the recommendation list from a recommender model trained on the original training data, and the recommendation list from a model trained on the perturbed training data. Then, the two recommendation lists are compared for each test interaction as shown in Figure 1. If the two lists are the same, then we say that the model is stable to the perturbation; otherwise, the model is unstable.

Our approach requires a metric to differentiate the order of items between two lists. Standard next-item prediction metrics such as MRR, Recall, NDCG, and AUC are applicable to one list (by measuring the rank of the ground-truth next item in the list). Extensions of these metrics, e.g., the difference in MRR, are not appropriate since those metrics can remain unchanged even if the rank list is drastically different, but the ground-truth item's rank remains similar. This happens in practice (see Figure 3(b)). Thus, we introduce a formal metric to quantify the stability of recommender systems, namely *Rank List Sensitivity* (RLS), which measures the similarity in the rank lists generated in the presence versus absence of perturbations. We employ two metrics to measure RLS, namely Rank-Biased Overlap (RBO) [69] and Jaccard Similarity [30]. RBO measures similarity in the order of items in two lists, while the Jaccard score highlights the overlap in the top-K items without considering their order. Higher scores in both metrics are better.

We introduce two training data perturbation methods to measure the stability of recommender systems: random perturbations and CASPER perturbations. Random perturbations select one interaction out of all training data interactions randomly for perturbations.

Table 1: Comparison of our proposed method (CASPER) against existing methods to measure perturbation and model stability.

	CASPER (Proposed)	Rev.Adv. [63]	LOKI [81]	S-attack [18]	PoisonRec [58]	CF-attack [40]	RL-attack [7]	LOO-user [6]
Deep Sequential Recommendation	✓	✓	✓		✓		✓	
Training Data Perturbation	✓	✓	✓	✓	✓	✓	✓	✓
Gray- or Black-box Perturbation	✓	✓	✓		✓			
Interaction-level Perturbation	✓							
Investigating Model Stability	✓							✓

Using random perturbations, we can measure the model sensitivity caused due to arbitrary errors and noise. On the other hand, CASPER is designed to identify an interaction whose perturbation can introduce higher instability in recommendations than random perturbations. Such interaction reveals model vulnerabilities that can potentially be exploited by adversaries to manipulate the recommender system. To find the deliberate perturbation, we hypothesize a cascading effect by creating an interaction-to-interaction dependency graph. Then, CASPER perturbs an interaction with the largest number of descendants in the graph, which leads to significant changes in the generated recommendations. CASPER is fast and scalable to the dataset size and does not require model parameters or gradients to identify the perturbation.

Experimentally, we first investigate the sensitivity of models to random perturbations. We show that the recommender models are sensitive to random interaction perturbations. Even *one* random interaction perturbation drastically changes the entire rank lists of items for *all* users. This is shown as low RBO scores (lower than 1.0 score means the rank list has changed) of four recommendation models on Foursquare (Figure 2(a)) and all four datasets (shown later in Figure 3), and as low top-10 Jaccard scores (Figure 7). We underline that the instability of the models occurs due to the data change, not the training randomness (e.g., different random seed, initialization, etc.), since we remove all the randomness during the training to focus solely on the effect of training data perturbation.

Next, we compare CASPER with five training data perturbation algorithms. We show that CASPER identifies a perturbation to be made that reveals higher sensitivity in recommendation models compared to existing methods across datasets. Importantly, we find that CASPER identifies an interaction whose perturbation results in low-accuracy user groups being more impacted as per model sensitivity — the top-10 Jaccard scores are lower for low-MRR users than for high-MRR users (see Figure 2(b)). Since the item ranking in the recommendation list has a significant impact on user satisfaction [53], if the recommendation is low-quality and unstable, the user satisfaction and engagement can be dramatically reduced, and it may result in user dropout. We provide the repository of our dataset and code used in the paper for reproducibility[1].

2 RELATED WORK

Data Perturbation in Recommender Systems. Our work is broadly related to a stream of research on perturbations in deep recommender systems [4, 10, 11, 16, 18, 19, 43, 44, 58, 72, 73, 80, 82, 83]. Much of this work has generated perturbations that alter the

[1] https://github.com/srijankr/casper

rank of target item(s) (see Table 1). These methods highlight the vulnerability of specific recommendations. However, they provide incomplete stability since they focus on specific target items, rather than the entire or top-K rank lists for all users.[2] Furthermore, they are not appropriate as baselines because they are not applicable for interaction-level perturbations or work only on multimodal recommenders [4, 11, 44] and matrix factorization-based models [18, 72, 73]. Some CF-based [40] and RL-based [7] recommender systems, provide untargeted perturbations for recommender systems that reduce the model's prediction accuracy considerably. However, those methods do not work on our perturbation setting since they provide user- or item-level perturbations instead of interaction-level or focus on degrading the model's prediction accuracy without altering the rank lists of all users.

Data Perturbation in Other Domains. Many input perturbation methods [20, 21, 47, 48, 57, 60, 84] have been developed for image classification. These methods cannot be directly applied to recommender systems due to complexities of sequential data (e.g., discrete input data and temporal dependency between interactions). Many data perturbation algorithms [8, 24, 37, 49, 66] for natural language processing (NLP) have been proposed. We cannot employ them directly for our setting since they either are targeted perturbations, have different perturbation levels (e.g., word or embedding modifications), or cannot model long sequential dependencies.

Stability & Multiplicity in Machine Learning. Our work is also related to a stream of work on stability and multiplicity in machine learning [3, 6, 12, 15, 27, 38, 45, 52, 56, 68]. Recent work in this area has shown that datasets can admit multiple nearly-optimal solutions that exhibit considerable differences in other desirable characteristics (e.g., predictions on specific data points, behavior to model shifts, counterfactual explanations). For instance, Black and Fredrikson [6] study data multiplicity caused by inserting or removing a single user ("leave-one-out") on several ML models. Marx et al. [45] demonstrate the potential fairness issue in recidivism prediction problems. While the majority of papers focus on the model multiplicity in classification models, they do not study the stability in recommender systems caused by data perturbations.

3 PRELIMINARIES

We consider a sequential recommendation task, where a recommender model $\mathcal{M} : X \rightarrow R^{X}_{\mathcal{M}}$ is trained to learn users' behavioral patterns from a sequence of their actions. A trained model $\mathcal{M}$ generates a rank list of all items $R^{X_k}_{\mathcal{M}}$ that a user may interact with given a test interaction $X_k \in X_{test}$. Items are ordered in terms of the likelihood of user interaction, and the system shows the top-K items from the rank list $R^{X_k}_{\mathcal{M}}[1:K]$ to each user. We denote the set of users and items as U and I, respectively. We study the sensitivity of four methods to train a sequential recommendation model:

- **LSTM [26]:** given a sequence of items, it predicts the next item via Long Short-Term Memory (LSTM).
- **TiSASRec [41]:** a recent self-attention based model that predicts the next item using the relative time intervals and absolute positions among previous items.
- **JODIE [36]:** a coupled RNN-based recommendation model which predicts the next item via RNNs to learn user and item embeddings.
- **LatentCross [5]:** a gated recurrent unit (GRU) [9] based model which uses contextual features, like time difference between interactions. This model is used in YouTube [5].

Table 2: Recommendation datasets used in Sections 5, 6, 7.

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
Wikipedia	1,914	1,000	142,143	Wikipedia page edit history
Reddit	4,675	953	134,489	Subreddit posting history

3.1 Datasets

We use four recommendation datasets from diverse domains summarized in Table 2. In each dataset, we filter out users with fewer than 10 interactions.

- LastFM [22, 25, 31, 39, 55] includes the music playing history of users represented as (user, music, timestamp).
- Foursquare [74, 76–78] is a point-of-interest dataset represented as (user, location, timestamp).
- Wikipedia [2, 14, 35, 36, 42, 50, 51] contains the edit records of Wikipedia pages represented as (user, page, timestamp).
- Reddit [1, 14, 34, 36, 42, 51] includes the posting history of users on subreddits represented as (user, subreddit, timestamp).

3.2 Next-Item Prediction Metrics

The dataset-level performance of a sequential recommendation model is evaluated in a next-item prediction task by calculating the rank of the ground-truth item among all items, averaged over all test interactions. Two metrics are widely used: (i) Mean Reciprocal Rank (MRR) [65]; (ii) Recall@K (typically K=10) [23, 36]. Both metrics lie between 0 and 1, and higher values are better. We refer to these two metrics as *next-item metrics* as they provide average statistics of the ranks of ground-truth next items.

4 MEASURING RANK LIST SENSITIVITY

We create a framework to measure the stability of recommendation systems against perturbations.

Procedure. First, we train a recommendation model $\mathcal{M}$ with the original data without perturbations, and it generates one ranked recommendation list $R^{X_k}_{\mathcal{M}}$ for each test interaction X_k in test data X_{test}, where k indicates an index of an interaction. Second, we train another recommendation model $\mathcal{M}'$ perturbed training data, and it generates ranked recommendation lists $R^{X_k}_{\mathcal{M}'}, \forall X_k \in X_{test}$.

We measure the similarity of recommendations for each test example X_k by comparing the two recommendation lists $R^{X_k}_{\mathcal{M}}$ and $R^{X_k}_{\mathcal{M}'}$. We devise Rank List Sensitivity (RLS) metrics to measure the similarity (described in the next paragraph). Then, we average the individual RLS score across all $X_k \in X_{test}$. If the model is perfectly stable, then $R^{X_k}_{\mathcal{M}}$ and $R^{X_k}_{\mathcal{M}'}$ should be identical $\forall X_k \in X_{test}$, and the average RLS value should be maximized.

We repeat the above process multiple times with different random seeds (to average the impact of individual experiments), and report average values of RLS metrics across different runs. The average RLS value quantifies the model stability.

Rank List Sensitivity Metrics. To measure the stability of a recommendation model $\mathcal{M}$, we need metrics that can compare the

[2] Setting all (top-K) items as targets can be inaccurate and computationally expensive.

similarity between recommendation lists generated with versus without perturbations, i.e., $R^{X_k}_{\mathcal{M}'}$ versus $R^{X_k}_{\mathcal{M}}$. Standard next-item prediction metrics (described in Section 3.2) only measure the rank of the ground-truth next item in one recommendation list. Extensions of these metrics, e.g., the difference in MRR or Recall, to measure similarity are not appropriate since these metrics can remain unchanged if the ground-truth item's rank is the same in $R^{X_k}_{\mathcal{M}}$ and $R^{X_k}_{\mathcal{M}'}$, even though the positions of the other items in the two rank lists are drastically different. This happens in practice – see Figure 3(b), where the difference between MRR and Recall values of recommendation models $\mathcal{M}$ and $\mathcal{M}'$ are almost identical. To compute the list similarity accurately, we need to measure how a perturbation impacts the order of *all items* across two recommendation lists. Thus, we need metrics that are sensitive to differences in the positions of all items, not only the ground-truth item.

We introduce a formal metric called *Rank List Sensitivity (RLS)* to quantify the stability of recommender systems by comparing the items and their ranking in two lists (or two top-K lists). Mathematically, RLS metrics of a model $\mathcal{M}$ against input perturbation are defined by the following:

$$RLS = \frac{1}{|X_{test}|} \sum_{\forall X_k \in X_{test}} sim(R^{X_k}_{\mathcal{M}}, R^{X_k}_{\mathcal{M}'})$$

where $sim(A, B)$ is a similarity function between two rank lists A and B. We use the following two similarity functions in this paper. (1) **RBO (Rank-biased Overlap):** RBO [69] measures the similarity of orderings between two rank lists $R^{X_k}_{\mathcal{M}'}$ and $R^{X_k}_{\mathcal{M}}$. RBO lies between 0 and 1. Higher RBO means the ordering of items in the two lists is similar. For reference, the RBO between two randomly-shuffled rank lists is approximately 0.5. RBO is more responsive to similarities in the top part of two rank lists, meaning that it imposes higher weights on the top-K items. This property distinguishes RBO from other measures like Kendall's Tau [32]. RBO of two rank lists A and B with $|I|$ items is defined as follows.

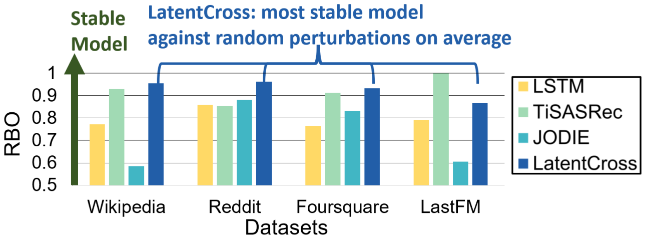
$$RBO(A, B) = (1 - p) \sum_{d=1}^{|I|} p^{d-1} \frac{|A[1:d] \cap B[1:d]|}{d}$$

where p is a tunable parameter (recommended value: 0.9).
(2) **Top-K Jaccard similarity**: The Jaccard similarity [30]

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

is a normalized measure of similarity of the contents of two sets A and B. We use it to measure the similarity of items in the top-K recommendation lists generated with and without perturbations, i.e., $R^{X_k}_{\mathcal{M}'}[1:K]$ and $R^{X_k}_{\mathcal{M}}[1:K]$. The Jaccard score ranges from 0 to 1, and is agnostic to the ordering of items. A model is stable if its Jaccard score is close to 1. In all experiments, we set K = 10 to compare the top-10 recommendations [23, 36].

Top-K Jaccard metric can be useful for the industry due to its fast computation compared to RBO; RBO can be used for detailed analyses of the model stability since it focuses on full ranked lists.



(a) Random leave-one-out (LOO) perturbation

	Next-item Metric 1: MRR			Next-item Metric 2: Recall@10		
Model	Without Perturbation	With Perturbation	Difference	Without Perturbation	With Perturbation	Difference
LSTM	0.8250	0.8241	**0.0009**	0.8976	0.8968	**0.0008**
TiSASRec	0.8212	0.8212	**0.0000**	0.8963	0.8969	**0.0006**
JODIE	0.8264	0.8257	**0.0007**	0.8860	0.8828	**0.0032**
LatentCross	0.8060	0.8056	**0.0004**	0.8709	0.8708	**0.0001**

(b) Next-item metrics of models against random LOO perturbation

Figure 3: (a) *Stability of four recommendation models against random LOO perturbation.* Existing models exhibit unstable predictions since RBO scores after the perturbation are low. (b) *Impact of random LOO perturbation on next-item predictions of recommendation models.* The differences in metrics with and without perturbations are marginal.

5 STABILITY AGAINST RANDOM PERTURBATIONS

In this section, we investigate the stability of recommendation models against random perturbations.

Interaction-level Perturbations. We measure the stability of a model with respect to arbitrary errors and noise through *minimal random perturbations*. These perturbations change one randomly-chosen sample in the training data – i.e., an interaction of a single user rather than all interactions of a user or an item. In particular, an interaction is either deleted (leave-one-out), inserted, or the interaction's item is replaced with another random item.

Experimental Setup. Our goal is to test the stability of diverse recommendation models against a random interaction perturbation. We use the first 90% of interactions of each user for training the recommendation model, and the rest are used for testing, which is a common setting used in several papers [17, 28, 46, 67]. For each model, we use the hyperparameters mentioned in their original publications. Other hyperparameters are set as follows: the maximum training epoch is set to 50, a learning rate is set to 0.001, and the size of the embedding dimension is set to 128. For LSTM and TiSASRec, the maximum sequence length per user is set to 50.

Procedure to Measure Stability. We follow the procedure described in Section 4 and use the two RLS Metrics to measure the stability of recommendation models against random perturbations.

Findings. We present the RBO scores of four recommendation models on Foursquare against random leave-one-out perturbation in Figure 3(a). We observe that all four recommendation models exhibit low RBO scores on all datasets, ranging from 0.75 to 0.95 in most cases, while sometimes dropping below 0.6. Recall that since the RBO score between two randomly-shuffled rank lists is approximately 0.5, it shows that the drop of RBO caused by perturbations is meaningful, but the rank list does not change randomly, which is expected. Similar drops are observed for top-10 Jaccard similarity and in the case of insertion and replacement perturbations. Insertion and replacement perturbation results are excluded due to space limitation. Thus, we observe the instability of existing models against even minor random perturbation. Notably, perturbation of a user's interaction leads to drastic changes in the recommendations of unrelated users.

Comparing the four models, LatentCross has the highest RBO in most cases against random perturbations. This indicates that LatentCross is the most stable model against random perturbations.

Controlling for Training Randomness while Measuring Model Stability. Other research has found that randomness during the training (e.g., random initialization, mini-batch shuffling, etc.) can generate different models and predictions in machine learning [3, 45]. Thus, in all our experiments (including the ones above), we specifically test the effect of the input data perturbation on model stability by controlling all other randomness (e.g., fixing the random seed and initialization). During a single run, we train two recommendation models $\mathcal{M}$ and $\mathcal{M}'$ (before and after perturbations) using *the exact same settings without any training randomness*. In other words, if there is no perturbation, the trained models $\mathcal{M}$ and $\mathcal{M}'$ and their outputs will be identical in every way.

Impact of Perturbations on Next-Item Prediction Metrics: We find that the trained recommender models with and without perturbations have similar dataset-level performance metrics (both have almost identical MRR and Recall scores), as shown in Table 3(b). However, the generated recommendation lists are drastically different, as indicated by the low RBO and Jaccard scores. This shows that multiple equivalent models can be trained that have similar dataset-level metrics, but provide conflicting recommendations. Similar findings have been made in other prediction settings [45].

One may wonder that if the dataset-level metrics are the same, is there any concern if the rank lists vary? We argue that this is indeed a matter of concern due to the following three reasons:
(a) Since several equivalent models generate different predictions, the specific recommendations, e.g., which drug to administer or which treatment procedure to follow, can vary depending on which model is used. It is important for the algorithm designer and the end-user to know that if the recommendation for a certain user can be easily changed by unrelated minor perturbations, then perhaps none of the recommendations should be followed for that user.
(b) Since multiple recommender models exist with equivalent next-item prediction performance, then how can the algorithm designer decide which model to deploy? We argue that given comparable models, stabler recommender models should be used.
(c) Our work highlights the importance of "beyond-accuracy" metrics (e.g., RLS metrics) given that different recommender models vary in their stability with respect to the RLS metrics.

Why are models unstable against minimal random perturbations? Only one interaction over one million interactions (size of the datasets used) is perturbed. Yet, it changes the rank lists and top-10 recommendation lists of all users. Why is there such a profound effect? This is due to two reasons.
(1) The slight change in training data leads to changes in the parameters of a trained recommendation model $\mathcal{M}$. Say an interaction in a mini-batch m was perturbed. When processing m, model parameters $\Theta(\mathcal{M})$ will be updated differently during training (compared to when there is no perturbation). The changes in $\Theta(\mathcal{M})$ will affect the updates in later mini-batches. The differences will further cascade and multiply over multiple epochs. Thus, with perturbations, the final $\Theta(\mathcal{M})$ will be different from the ones obtained without perturbations, which can result in different rank lists.
(2) The model $\mathcal{M}$ is trained to accurately predict only the ground-truth next item as high in the rank list as possible (ideally, rank 1). However, $\mathcal{M}$ is not trained to optimize the positions of the other items in the rank list. Thus, the ordering of all except the ground-truth next item is highly likely to change due to input perturbation.

6 STABILITY AGAINST CASPER PERTURBATION

While random perturbations show the model instability introduced due to arbitrary errors and noise, it is essential to find perturbations that can lead to even higher instability, which helps understand the lowest stability exhibited by a model. Adversaries can potentially exploit such perturbations to conduct untargeted attacks and make the recommendations unstable for all users. Thus, in this section, we ask: ***which interaction should be perturbed to yield maximum instability in a recommendation model?*** We aim to find perturbations that maximally change the rank lists $R^{X_k}_{\mathcal{M}}$ compared to $R^{X_k}_{\mathcal{M}'}$, $\forall X_k \in X_{test}$. As before, we will consider *minimal interaction-level perturbations*, allowing one interaction to be perturbed. *Three types of perturbations* can be made: leave-one-out (LOO), insertion, and replacement.Due to space constraints, we will highlight LOO perturbation results as other perturbations yield similar model instability. Finally, we will consider *gray-box perturbations* — we assume access to training data and some model information such as the maximum sequence length of past user actions that the recommendation model uses to make predictions. Note that we do *not* require any details of the recommendation model such as the model's architecture, parameters, or gradients [19, 29, 80].

6.1 Perturbing Interactions from Different Timestamps

A brute-force technique that tests the impact of every interaction perturbation on model stability is computationally prohibitive due to the need to retrain the model after each perturbation. To find an effective perturbation in a scalable manner, we first investigate the impact of perturbations in different positions in the training data.

We take inspiration from an idea from temporal recommendation models [5, 26, 36], where mini-batches $\mathcal{B} = \{B_1, \ldots, B_T\}, B_1 \cup \cdots \cup B_T = X_{train}$ are created in temporal order (i.e., first P interactions in the first batch B_1, and so on). In such models, earlier batches contain training interactions with early timestamps, and perturbing an interaction in the earlier batches is equivalent to perturbing an interaction with early timestamps. Since we saw in the case of random perturbations that the impact of perturbations on model parameters can cascade, we ask: ***how does perturbing interactions from different timestamps impact model stability?***

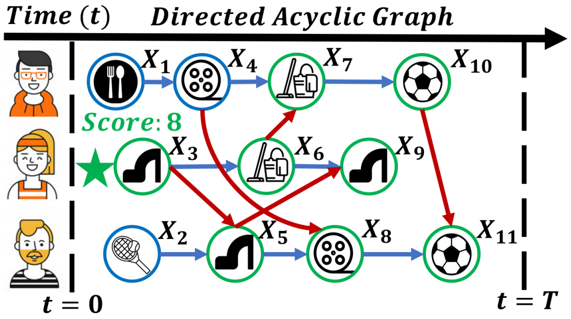
We devise and compare three following heuristic perturbations: an *Earliest-Random perturbation*, where the first interaction of a randomly selected user is perturbed, a *Latest-Random perturbation*, where the last interaction of a randomly selected user is perturbed,

Algorithm 1: CASPER: interaction-level perturbation based on cascading effect

```
Input  : Training interaction data $X_{train}$, users and items $U$ and $I$, training interaction sequences $X^u$ and $X^i$ (sorted by timestamp) for
         each user $u$ and item $i$
Output : Perturbed training data $X_{perturbed}$
1  Initialize an interaction-to-interaction directed acyclic graph (IDAG) $G$ with all training interactions $X_k \in X_{train}$ as nodes
2  for each user $u \in U$ do                                        ▷ Creating edges in the IDAG $G$
3      for $k \in [1, 2, \ldots, |X^u| - 1]$ do                     ▷ Adding edges between consecutive interactions of $u$
4          Create an edge from $X^u_k$ to $X^u_{k+1}$ in $G$
5  for each item $i \in I$ do                                        ▷ Creating edges in the IDAG $G$
6      for $k \in [1, 2, \ldots, |X^i| - 1]$ do                     ▷ Adding edges between consecutive interactions of $i$
7          Create an edge from $X^i_k$ to $X^i_{k+1}$ in $G$
8  for each interaction $X_k = (u, i, t) \in X_{train}$ do          ▷ Compute cascading scores $score(X_k)$, $\forall X_k \in X_{train}$
9      if $Indegree(X_k) == 0$ then
10         Perform breadth-first search (BFS) starting from $X_k$ to find all the descendants of $X_k$ in $G$
11         $score(X_k) \longleftarrow$ total number of descendants of $X_k$ in $G$
   ▷ Perturb the interaction with the highest cascading score
12 To obtain new perturbed training data $X_{perturbed}$, perturb the interaction $X_{opt} = \underset{\forall X_k \in X_{train}}{\arg\max} (score(X_k))$
```

Perturbations	LOO	Replacement	Insertion
Earliest-Random	**0.8563**	**0.8421**	**0.7396**
Random	0.9335	0.9493	0.9350
Latest-Random	0.9608	0.9850	0.9641

(a) Impact of perturbing interactions from different timestamps.



(b) An IDAG corresponding to Figure 1.

Figure 4: (a) Average RBO scores of perturbing interactions from different positions in the training data. Earliest-Random perturbation produces lower RBO than Random and Latest-Random perturbations. (b) An IDAG corresponding to the interaction data in Figure 1. Blue and red edges indicate user- and item-sharing adjacent interactions, respectively. Green-colored nodes (interactions) show all descendants (including itself) of an interaction X_3. The cascading score of $X_3 = 8$, which is its number of descendants.

and a *Random perturbation*, where a random training interaction is perturbed. We test this cascading effect on LatentCross model since it was the most stable against random perturbation.

We use the RBO metric to measure RLS caused by these perturbations on the LatentCross model and Foursquare dataset (the hardest-to-predict dataset as per next-item metrics). We perform each perturbation 10 times (randomly perturbing one interaction only each time). The resulting RBO score distributions are compared using the Wilcoxon signed-rank test [71].

The RBO scores are shown in Table 4(a). Earliest-Random perturbation leads to the lowest RBO score in all three types of perturbations, i.e., LOO, replacement, and insertion (all p-values <0.05). We also observe that between Random and Latest-Random, the former has lower RBOs. These findings show that perturbing earlier timestamp interactions leads to higher instability in recommendations. Since this happens due to the cascading impact of model parameter changes over mini-batch updates, we call this a "*cascading effect*".

6.2 CASPER: Interaction-level Perturbation based on Cascading Effect

Now, we leverage the cascading effect to propose a new perturbation, named CASPER (Cascade-based Perturbation).

To approximate the impact of perturbing an interaction X_k, we define a *cascading score* of X_k as the number of training interactions that will be affected if X_k is perturbed. Inspired by temporal recommendation models [5, 26, 36], we create an interaction-to-interaction dependency graph, which encodes the influence of one interaction on another. Then, we approximate the cascading score of interaction X_k as the number of descendants of X_k in this graph. CASPER aims to identify the training interaction which has the highest cascading score, since its perturbation would maximize the cascading effect. Algorithm 1 shows the key steps of the method.

Creating the interaction-to-interaction dependency DAG: We create a graph-based technique to approximate an interaction's cascading score without retraining the recommendation model. We first construct an interaction-to-interaction dependency directed acyclic graph (IDAG; lines 1-7 in Algorithm 1), where nodes are training interactions and directed edges represent which interaction influences another. The edge encodes the dependency that if the k^{th} interaction of user u (or item i) is perturbed, it will influence the $k + 1^{th}$ interaction of user u (or item i). The IDAG corresponding to the training interactions from Figure 1 is presented in Figure 4(b).

Two nodes in the IDAG are connected by a directed edge if they are either consecutive interactions of the same user (e.g., X_1 and X_4) or of the same item (e.g., X_3 and X_5). A directed edge must follow the temporal order from early to later timestamp. No edges are present between nodes with the same timestamp. Thus, each node has at most two outgoing edges (first to the next interaction of the user and second to the next interaction of the item). If the recommendation model has a maximum sequence length (L), the IDAG is constructed only with the latest L interactions of each user.
Calculating the cascading score in IDAG: The cascading score of a node X_k is approximated as the total number of descendants of X_k in the IDAG. Descendants of a node X_k in the IDAG are defined as all the nodes reachable from X_k by following the outgoing edges in the IDAG. For example, in Figure 4(b), X_3 has 8 descendants (including itself), the highest among all nodes. By definition, a node's parent will have a higher cascading score than the node itself. Hence, we accelerate the computation by calculating the cascading scores of zero in-degree nodes only (lines 8-11 in Algorithm 1). Finally, CASPER perturbs the node with the highest cascading score since it would maximize the cascading effect (line 12 in Algorithm 1).

We have theoretically and experimentally shown that CASPER scales near-linearly to the dataset size (Section 6.3 and Figure 8(b)).

6.3 Complexity Analyses of CASPER

We analyze the time and space complexities of CASPER. We assume the maximum sequence length of a model is L.
Time complexity. CASPER first trains and tests a given recommendation model Θ with original input data, which takes $O(\mathcal{T}(\Theta))$, where $\mathcal{T}(\Theta)$ is the time complexity of Θ. After that, CASPER constructs the IDAG which takes $O(|U|L)$ where $|U|$ is the number of users. Computing cascading scores of zero in-degree nodes in the IDAG, which takes $O(Z|U|L)$ where Z is the number of zero in-degree nodes in the IDAG. Perturbing an interaction with the highest cascading scores takes $O(Z)$. Finally, CASPER retrains the model Θ with perturbed data and computes RLS metrics, which takes $O(\mathcal{T}(\Theta) + N_{test}|I|)$ since RBO should be calculated with all items $|I|$, where N_{test} is the number of test interactions. The final time complexity of CASPER is $O(\mathcal{T}(\Theta) + N_{test}|I| + Z|U|L)$.
Space complexity. The first step of CASPER is training and testing a deep sequential recommendation model Θ with original input data, which takes $O(\mathcal{S}(\Theta) + N_{test}|I|)$ space since we need to store original rank lists for all test interactions, where $\mathcal{S}(\Theta)$ is the space complexity of Θ. After that, CASPER constructs the IDAG which takes $O(|U|L)$ space. The next step is computing cascading scores of zero in-degree nodes in the IDAG, which takes $O(|U|L)$ space. Finally, CASPER retrains the model Θ with perturbed data and computes RLS metrics, which takes $O(\mathcal{S}(\Theta) + N_{test}|I|)$ space. The final space complexity of CASPER is $O(\mathcal{S}(\Theta) + N_{test}|I| + |U|L)$.

7 EXPERIMENTAL EVALUATION OF CASPER

In this section, we evaluate CASPER by the following aspects.
(1) **Stability of Recommendation Models against Diverse Perturbations (Section 7.2).** How stable are existing recommender systems against CASPER and baseline perturbations?
(2) **Impact of Perturbations on Different Users (Section 7.3).** Are there any user groups that are more susceptible and sensitive to input data perturbations?
(3) **Impact of the Number of Perturbations (Section 7.4).** Is the performance of CASPER proportional to the number of perturbations allowed on the dataset?
(4) **Running Time Analysis (Section 7.5).** Does the running time of CASPER scale with the dataset size?

7.1 Experimental Settings

7.1.1 Datasets. We use the four standard datasets introduced in Section 5. LastFM is a widely used recommendation benchmark dataset [22, 31, 39, 55], Foursquare is broadly utilized for point-of-interest recommendations [74, 76–78], and Wikipedia and Reddit are popular for social network recommendations [14, 36, 42, 51]. We select these datasets for experiments because (a) they come from diverse domains, thus ensuring generalizability, and (b) the timestamps of interactions reflect when the corresponding activities happened (as opposed to Amazon review datasets where a review is posted much after a product is purchased, or MovieLens review dataset where a review is posted much after a movie is watched).

7.1.2 Baseline Methods. To the best of our knowledge, there are no interaction-level perturbation methods for existing recommendation models. Therefore, we create strong baselines and two state-of-the-art methods based on the broader literature as follows:
- **Random perturbation:** It randomly chooses an interaction for perturbation among all training interactions.
- **Earliest-Random perturbation:** It randomly chooses an interaction for perturbation among the first interactions of all users in the training data.
- **Latest-Random perturbation:** It randomly chooses an interaction for perturbation among the last interactions of all users in the training data.
- **TracIn [54] perturbation:** It chooses the most important training interaction for perturbation, defined in terms of reducing the model's loss during training. We use an influence estimator TracIn [54] that utilizes loss gradients from the model saved at every T epoch to compute interaction importance.
- **Rev.Adv. [63] perturbation:** It inserts a fake user with interactions crafted via a bi-level optimization problem for perturbations. To adapt it for our leave-one-out (LOO) and replacement perturbation settings, we first find the most similar user in the training data to the fake user, and perform LOO or item replacement of the earliest or random interaction of that user, respectively. Therefore, we create two versions of Rev.Adv. – Rev.Adv. [63] (random) and Rev.Adv. [63] (earliest), which indicates the method chooses a random or earliest interaction of a user for perturbation, respectively.

Note that we do not include baselines that work only on multimodal recommenders [4, 11, 44] and matrix factorization-based models [18, 72, 73] as these are not applicable to our setting. We have also not included baselines that have shown similar or worse performance [58, 80, 81] compared to the above baselines, particularly compared to Rev.Adv. [63]. In replacement and insertion perturbations, the new item can be selected using three different strategies: selecting an item randomly, selecting the most popular item, or selecting the least popular (i.e., unpopular) item.

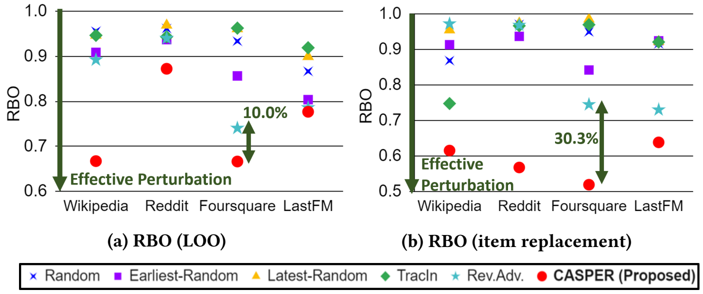
Table 3: *Effectiveness of perturbations on Foursquare dataset.* We find instability of existing recommendation models measured by the RBO metric against LOO (left) and item replacement perturbations (right). All RBO scores are lower than 1.0. The best perturbation in each column is colored blue, and the second best is light blue, in terms of achieving the lowest RBO score.

a LOO perturbation comparison using RBO

Model / Perturbations	LSTM	TiSASRec	JODIE	LatentCross
Random	0.7799	0.9117	0.8316	0.9335
Earliest-Random	0.7876	0.8776	0.8211	0.8563
Latest-Random	0.7763	0.8515	0.8420	0.9608
TracIn [54]	0.7733	0.8545	0.8713	0.9625
Rev.Adv. [63] (random)	0.7798	0.8955	0.8491	0.9317
Rev.Adv. [63] (earliest)	0.7787	0.8911	**0.7185**	0.7403
Proposed method				
CASPER	**0.7709**	**0.8450**	0.7896	**0.6662**

b Item replacement perturbation comparison using RBO

Model / Perturbations	LSTM	TiSASRec	JODIE	LatentCross
Random	0.7795	0.9143	0.9414	0.9493
Earliest-Random	0.7743	0.8886	0.8871	0.8421
Latest-Random	0.7814	0.8553	0.8989	0.9850
TracIn [54]	0.7934	0.8520	0.9280	0.9696
Rev.Adv. [63] (random)	0.7856	0.8782	0.9159	0.9538
Rev.Adv. [63] (earliest)	0.7747	0.9375	0.8257	0.7449
Proposed method				
CASPER (random)	0.7665	0.8482	0.6691	0.6065
CASPER (popular)	**0.7557**	0.8477	0.6114	0.5435
CASPER (unpopular)	0.7615	**0.8471**	**0.5228**	**0.5193**



(a) RBO (LOO) **(b) RBO (item replacement)**

Figure 5: *Comparing perturbations on LatentCross model across all datasets.* CASPER shows the best perturbation performance.

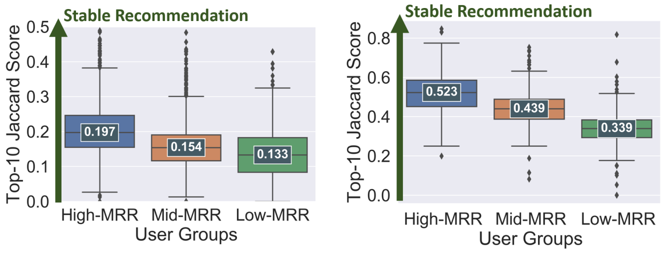
7.1.3 Recommendation Models. We use popular recommender models: LSTM [26], TiSASRec [41], JODIE [36], and LatentCross [5] described earlier to test the effectiveness of CASPER and baselines.

7.1.4 Experimental Setup. We follow the same experimental setup, as described previously in Section 5. Additionally, we use the following settings. We repeat all experiments multiple times and report average values of RLS metrics. To construct the IDAG for CASPER, we use all the interactions in JODIE and LatentCross. For LSTM and TiSASRec, we use the latest 50 interactions per user, as defined by the maximum sequence length in the original papers. To compute the influence of interactions in the TracIn perturbation, we take training loss gradients with respect to the last hidden layer. We save the loss gradients every 10 epochs and fix step sizes to the default learning rate of 0.001.

7.2 Stability of Recommendation Models against Diverse Perturbations

Perturbations of all models on Foursquare dataset. Table 3 compares the performance of all perturbation methods on all four recommendation models and Foursquare dataset (the hardest-to-predict in terms of next-item metrics), averaged over 3 repetitions. Each column highlights the **best** and **second-best** perturbation model, in terms of the lowest RBO score.

We observe the instability of all recommendation models against LOO and replacement perturbations. The RBO scores of all the recommendation models drop significantly below 1.0, indicating their low stability. CASPER achieves the best performance across all but one setting, where it performs the second best. It leads to the most reduction of RBO in most cases. CASPER shows lower variances of RLS values than those of baselines across different runs. We observe that CASPER is more effective on JODIE and LatentCross models, since the other two models (LSTM and TiSASRec) use maximum sequence lengths, which limit their interactions' cascading effects. In some cases, e.g., JODIE and LatentCross in item replacement, their resulting RBO drops close to 0.5, which is similar to the case of random shuffling of ranked lists. Similar observations hold with top-K Jaccard score and for insertion perturbations. It is also worth mentioning that Rev.Adv. (earliest) outperforms Rev.Adv. (random) in most cases, which also substantiates the cascading effect.



(a) Jaccard@10 on Foursquare **(b) Jaccard@10 on LastFM**

Figure 6: *Comparing impact of perturbations across user groups.* Users with low accuracy receive more unstable predictions when CASPER perturbation is applied, which can cause a user fairness issue. This plot is for LOO perturbation results on LatentCross model.

In *item replacement perturbation* (Table 3(b)), CASPER outperforms other methods in all cases. For CASPER, replacing the item with the least popular item is the most effective strategy among all the others. One possible reason is that the change in user embeddings and model parameters by using an unpopular item will be the highest. Injection of the unpopular item diversifies the user's interactions and embedding the most, and model parameters can be updated most differently. This major update will cascade to later interactions and change all users' recommendations drastically.

Perturbations on LatentCross model on all datasets. We further evaluate the effectiveness of CASPER versus baselines on the LatentCross model (the most stable model against random perturbations) across four datasets. The results are shown in Figures 5(a) and 5(b). We confirm unstable predictions of LatentCross against CASPER LOO and item replacement perturbations as per RBO.

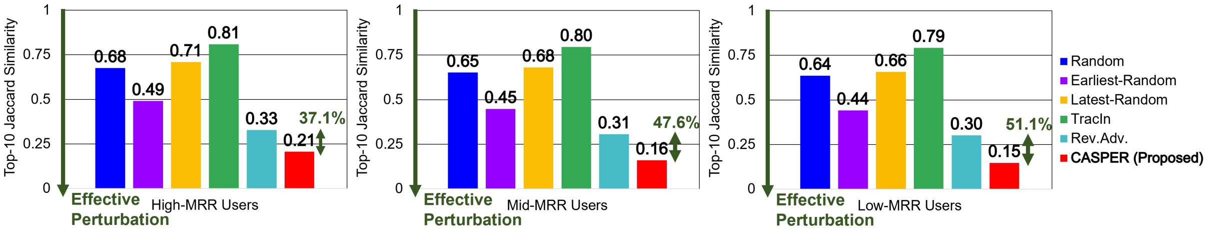


Figure 7: *Comparing different perturbation methods' impact across user groups.* We see LOO perturbation on LatentCross model and Foursquare dataset as per top-K Jaccard similarity (K=10), averaged over users with high-MRR (left), mid-MRR (middle), and low-MRR (right), respectively. Users with low accuracy suffer more from training data perturbations with any perturbation method. CASPER leads to the highest reduction of top-10 Jaccard similarity and outperforms all baselines across all user groups.

Top-K Jaccard metric and insertion perturbations also show similar results. Notably, CASPER outperforms all baselines across all datasets on the LatentCross model.

Across all datasets, Latest-Random baseline performs worse than the Random, which performs worse than the Earliest-Random, due to cascading effect. Similarly, all random perturbations have worse performance than advanced perturbations like CASPER.

7.3 Impact of Perturbations on Different Users

To investigate the differential impact of training data perturbations on different users, we divide users into three groups: (1) High-MRR users, containing users who lie in the top 20% according to average MRR, (2) Low-MRR users, containing users with the lowest 20% average MRR, and (3) Mid-MRR users, which contains the remaining set of users. We contrast the average RLS of users across the three groups. Figures 6(a) and 6(b) compare the top-10 Jaccard scores across the three user groups on LatentCross model and two datasets (Foursquare and LastFM) against CASPER LOO perturbation. We discover that the trend of stability follows the accuracy trend – users with high accuracy receive relatively more stable predictions than the low-accuracy user group. This phenomenon highlights the relatively higher instability faced by users for which the model is already unable to make accurate predictions. This raises an aspect of unfairness across user groups. This highlights the need that practitioners should evaluate model stability across user groups before deploying models in practice.

Furthermore, we observe the same trend across different perturbation methods, as shown in Figure 7. Regardless of the perturbation method, low-MRR users experience lower stability compared to the other two groups. Notably, CASPER is able to generate the lowest stability across all user groups. Addressing the differential impact across user groups will be important to study in future work.

7.4 Impact of the Number of Perturbations

Intuitively, more perturbations in training data will cause higher instability of a model. To test the effect of the number of perturbations on CASPER, we increase the number of perturbations from 1 to 8 and check its LOO perturbation performance on LatentCross model and Foursquare dataset. CASPER selects k interactions with the highest cascading score when the number of perturbations is k. As shown in Figure 8(a), the performance of CASPER scales near-linearly with the number of perturbations. Replacement and insertion perturbations show similar trends.

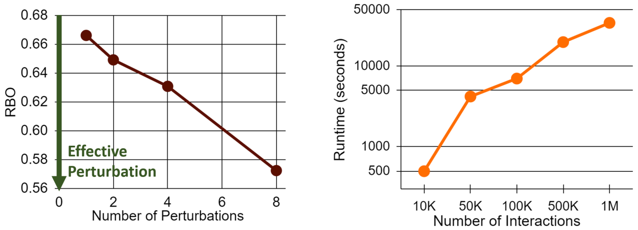


Figure 8: *(a) Perturbation scalability and (b) runtime* of CASPER.

7.5 Running Time Analysis

We vary the number of interactions in a dataset to test whether the runtime of CASPER is scalable to the input data size. Specifically, we measure the running time of LOO perturbation of CASPER on LatentCross model and LastFM dataset (the largest), while varying the number of interactions in the dataset from 10,000 to 1,000,000. Figure 8(b) shows CASPER scales near-linearly with the dataset size. This empirically validates the time complexity of CASPER (see Section 6.3), which is linear as per the total number of interactions.

8 CONCLUDING REMARKS

Our work highlights that recommendation models can exhibit instability to minor changes in their training data. These effects underscore the need to measure this instability, and to develop methods that are robust to such changes. The measures and methods developed in this paper are an initial step in this direction. In particular, CASPER depends on cascading effect which is inspired by temporal recommendation models, meaning that it may return solutions that are sub-optimal for methods that are not trained with temporally-ordered mini-batches.

Future work topics include: expanding CASPER to handle more complex perturbations, or to find more effective perturbations (e.g., interaction reordering) for other training regimes; and improving scalability of CASPER to handle very large interaction graphs (e.g., by creating approximations of cascading scores using a randomly-sampled interaction graphs, rather than the entire graph); developing methods that induce stability to data perturbations (e.g., via multi-objective learning aiming to accurately predict next items and preserve rank lists of a recommendation model simultaneously).

ACKNOWLEDGMENTS

This research is supported in part by Georgia Institute of Technology, IDEaS, and Microsoft Azure. S.O. was partly supported by ML@GT, Twitch, and Kwanjeong fellowships. We thank the reviewers for their feedback.

REFERENCES

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

[2] 2020. Wikipedia edit history dump. https://meta.wikimedia.org/wiki/Data_dumps.

[3] Junaid Ali, Preethi Lahoti, and Krishna P Gummadi. 2021. Accounting for Model Uncertainty in Algorithmic Discrimination. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 336–345.

[4] 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 *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1094–1103.

[5] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[6] Emily Black and Matt Fredrikson. 2021. Leave-one-out Unfairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 285–295.

[7] 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 *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1669–1672.

[8] Alvin Chan, Yi Tay, Yew-Soon Ong, and Aston Zhang. 2020. Poison Attacks against Text Datasets with Conditional Adversarially Regularized Autoencoder. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 4175–4189.

[9] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[10] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 322–330.

[11] Rami Cohen, Oren Sar Shalom, Dietmar Jannach, and Amihood Amir. 2021. A Black-Box Attack Model for Visually-Aware Recommender Systems. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 94–102.

[12] Amanda Coston, Ashesh Rambachan, and Alexandra Chouldechova. 2021. Characterizing fairness over the set of good models under selective labels. In *International Conference on Machine Learning*. PMLR, 2144–2155.

[13] Ricardo João Cruz-Correia, Pedro Pereira Rodrigues, Alberto Freitas, Filipa Canario Almeida, Rong Chen, and Altamiro Costa-Pereira. 2009. Data quality and integration issues in electronic health records. In *Information discovery on electronic health records*. Chapman and Hall/CRC, 73–114.

[14] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Recurrent co-evolutionary latent feature processes for continuous-time recommendation. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 29–34.

[15] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification Presents Challenges for Credibility in Modern Machine Learning. *Journal of Machine Learning Research* (2020).

[16] Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. 2020. Taamr: Targeted adversarial attack against multimedia recommender systems. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 1–8.

[17] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*. 152–160.

[18] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*. 3019–3025.

[19] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 381–392.

[20] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

[21] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. 2019. Simple black-box adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2484–2493.

[22] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1569–1577.

[23] 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 *Fourteenth ACM Conference on Recommender Systems*. 53–62.

[24] Bing He, Mustaque Ahamad, and Srijan Kumar. 2021. Petgen: Personalized text generation attack on deep sequence embedding-based classification models. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 575–584.

[25] Balázs Hidasi and Domonkos Tikk. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 67–82.

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[27] Hsiang Hsu and Flavio du Pin Calmon. 2022. Rashomon Capacity: A Metric for Predictive Multiplicity in Probabilistic Classification. *arXiv:2206.01295* (2022).

[28] Haoji Hu and Xiangnan He. 2019. Sets2sets: Learning from sequential sets with neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1491–1499.

[29] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data Poisoning Attacks to Deep Learning Based Recommender Systems. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.

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

[31] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2019. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 447–455.

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

[33] Yaa A Kumah-Crystal, Claude J Pirtle, Harrison M Whyte, Edward S Goode, Shilo H Anders, and Christoph U Lehmann. 2018. Electronic health record interactions through voice: a review. *Applied clinical informatics* 9, 03 (2018), 541–552.

[34] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community interaction and conflict on the web. In *Proceedings of the 2018 world wide web conference*. 933–943.

[35] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. 2015. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 607–616.

[36] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[37] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight Poisoning Attacks on Pretrained Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2793–2806.

[38] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. 2022. Diversify and Disambiguate: Learning From Underspecified Data. *arXiv:2202.03418* (2022).

[39] 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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2073–2083.

[40] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* 29, 1885–1893.

[41] 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.

[42] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic graph collaborative filtering. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 322–331.

[43] Fang Liu and Ness Shroff. 2019. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*. PMLR, 4042–4050.

[44] Zhuoran Liu and Martha Larson. 2021. Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start. In *Proceedings of the Web Conference 2021*. 3590–3602.

[45] Charles Marx, Flavio Calmon, and Berk Ustun. 2020. Predictive multiplicity in classification. In *International Conference on Machine Learning*. PMLR, 6765–6774.

[46] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring data splitting strategies for the evaluation of recommendation models. In *Fourteenth ACM Conference on Recommender Systems*. 681–686.

[47] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[49] Milad Moradi and Matthias Samwald. 2021. Evaluating the Robustness of Neural Language Models to Input Perturbations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 1558–1570.

[50] Manoj Niverthi, Gaurav Verma, and Srijan Kumar. 2022. Characterizing, Detecting, and Predicting Online Ban Evasion. In *Proceedings of the ACM Web Conference 2022*. 2614–2623.

[51] Shalini Pandey, George Karypis, and Jaideep Srivasatava. 2021. IACN: Influence-Aware and Attention-Based Co-evolutionary Network for Recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 561–574.

[52] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. On counterfactual explanations under predictive multiplicity. In *Conference on Uncertainty in Artificial Intelligence*. PMLR, 809–818.

[53] 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 *Proceedings of the 13th ACM Conference on Recommender Systems*. 3–11.

[54] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating Training Data Influence by Tracing Gradient Descent. *Advances in Neural Information Processing Systems* 33.

[55] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten De Rijke. 2019. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4806–4813.

[56] Xavier Renard, Thibault Laugel, and Marcin Detyniecki. 2021. Understanding Prediction Discrepancies in Machine Learning Classifiers. *arXiv:2104.05467* (2021).

[57] Sayantan Sarkar, Ankan Bansal, Upal Mahbub, and Rama Chellappa. 2017. UPSET and ANGRI: Breaking high performance image classifiers. *arXiv:1707.01159* (2017).

[58] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 157–168.

[59] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 3520–3532.

[60] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[61] Wencheng Sun, Zhiping Cai, Yangyang Li, Fang Liu, Shengqun Fang, and Guoyan Wang. 2018. Data processing and text mining technologies on electronic medical records: a review. *Journal of healthcare engineering* 2018 (2018).

[62] Huiyi Tan, Junfei Guo, and Yong Li. 2008. E-learning recommendation system. In *2008 International conference on computer science and software engineering*, Vol. 5. IEEE, 430–433.

[63] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems*. 318–327.

[64] 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), 171–201.

[65] Ellen M Voorhees et al. 1999. The trec-8 question answering track report.. In *Text Retrieval Conference*, Vol. 99. 77–82.

[66] Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed Data Poisoning Attacks on NLP Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 139–150.

[67] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[68] Jamelle Watson-Daniels, David C Parkes, and Berk Ustun. 2022. Predictive Multiplicity in Probabilistic Classification. *arXiv:2206.01131* (2022).

[69] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems* 28, 4 (2010).

[70] Martin Wiesner and Daniel Pfeifer. 2014. Health recommender systems: concepts, requirements, technical basics and challenges. *International journal of environmental research and public health* 11, 3 (2014), 2580–2607.

[71] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[72] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2021. Triple Adversarial Learning for Influence based Poisoning Attack in Recommender Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1830–1840.

[73] 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 *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1074–1083.

[74] 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 *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1245–1254.

[75] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative poisoning attack method against neural networks. *arXiv:1703.01340* (2017).

[76] Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 458–461.

[77] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 363–372.

[78] Quan Yuan, Gao Cong, and Aixin Sun. 2014. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 659–668.

[79] 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.

[80] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *Fifteenth ACM Conference on Recommender Systems*. 44–54.

[81] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical data poisoning attack against next-item recommendation. In *Proceedings of The Web Conference 2020*. 2458–2464.

[82] Hengtong Zhang, Y. Li, B. Ding, and Jing Gao. 2020. Practical Data Poisoning Attack against Next-Item Recommendation. In *TheWebConf*.

[83] 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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2154–2164.

[84] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*. 4480–4488.

[85] Dávid Zibriczky12. 2016. Recommender systems meet finance: a literature review. In *Proc. 2nd Int. Workshop Personalization Recommender Syst*. 1–10.