Variation Control and Evaluation for Generative Slate Recommendations

Shuchang Liu[1*], Fei Sun[2†], Yingqiang Ge[1], Changhua Pei[2], Yongfeng Zhang[1]
[1]Rutgers University, New Brunswick, NJ, USA [2]Alibaba Group, Beijing, China
[1]{shuchang.syt.liu, yingqiang.ge, yongfeng.zhang}@rutgers.edu [2]{ofey.sf, changhua.pch}@alibaba-inc.com

ABSTRACT

Slate recommendation generates a list of items as a whole instead of ranking each item individually, so as to better model the intra-list positional biases and item relations. In order to deal with the enormous combinatorial space of slates, recent work considers a generative solution so that a slate distribution can be directly modeled. However, we observe that such approaches—despite their proved effectiveness in computer vision—suffer from a trade-off dilemma in recommender systems: when focusing on reconstruction, they easily over-fit the data and hardly generate satisfactory recommendations; on the other hand, when focusing on satisfying the user interests, they get trapped in a few items and fail to cover the item variation in slates. In this paper, we propose to enhance the accuracy-based evaluation with slate variation metrics to estimate the stochastic behavior of generative models. We illustrate that instead of reaching to one of the two undesirable extreme cases in the dilemma, a valid generative solution resides in a narrow "elbow" region in between. And we show that item perturbation can enforce slate variation and mitigate the over-concentration of generated slates, which expand the "elbow" performance to an easy-to-find region. We further propose to separate a pivot selection phase from the generation process so that the model can apply perturbation before generation. Empirical results show that this simple modification can provide even better variance with the same level of accuracy compared to post-generation perturbation methods.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Generative Recommendation; Slate Recommendation; Conditional Variational Auto-Encoder

ACM Reference Format:
Shuchang Liu, Fei Sun, Yingqiang Ge, Changhua Pei, and Yongfeng Zhang. 2021. Variation Control and Evaluation for Generative Slate Recommendations. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449864

*This work was done when Shuchang Liu was an intern at Alibaba.
†Corresponding author.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449864

1 INTRODUCTION

In most recommender systems, items are naturally exposed to users as a slate, which usually contains a *fixed* number of items, e.g., a 1-by-5 list of recommended items, or a 2-by-2 block that can fit a mobile phone screen. This leads to the idea of slate recommendation, also known as exact-k recommendation [18, 42]. The problem is usually formalized as generating a slate of items such that certain expected user behavior (e.g., the number of clicks) is maximized. The challenge of this problem is that the number of possible slates is combinatorially large [44]. For example, for a system with n items, to generate a slate of k items, the possible number of slates will be $O(n^k)$, which is huge given that many recommender systems work on millions or even billions of items.

Traditional ranking-based recommendation models such as learning-to-rank (LTR) [7, 8, 17, 33, 37] first predicts the probability of user engagement on each candidate item, and then selects the top-ranked ones as the recommendation list. Despite its well-recognized effectiveness and scalability, this ranking and selection process is greedy in essence and neglects the fact that the user behavior on an item may be influenced by other (e.g., complementary or competitive) items exposed in the same list [29, 48], thus resulting in its sub-optimality. Furthermore, evidence has shown that one can improve the recommendation performance by taking into account the intra-list item relations in ranking [2, 8, 13, 18, 36, 48].

Recently, researchers have explored the possibility of solving this problem by directly generating the slate as a whole to break the limitation of ranking-based approaches. Many of the approaches are based on generative models such as Variational Auto-Encoders (VAE) [23, 28]. However, these generative models are stochastic in nature and their variational behavior may not produce satisfactory slate recommendations. For example, in the case of VAE-based models, the performance depends on a trade-off coefficient β [23]—the larger the β-value during training, the more the model is focused on encoding variation control against the data reconstruction accuracy. In terms of slate recommendation, this phenomenon diverges the generative results into one of the three cases:

- **Over-reconstruction**: when β is smaller than some lower threshold β^-, it tends to overfit the slate reconstruction on the training set. Though the resulting generated slates have extremely high variance, the model usually fails to generate satisfactory recommendations.
- **Over-concentration**: when β is larger than some upper threshold β^+, the model tends to choose from only a few prototypical slates that achieve satisfactory performance but fails to explore the variety of slates.
- **Elbow case**: when β is selected in an appropriate region (i.e., $\beta \in [\beta^-, \beta^+]$), it gives intermediate item variety and is

able to fulfill certain degree of user interests. We show that this transitional region is the most suitable for slate recommendation task. Unfortunately, this very setting usually lies in a narrow region (e.g. $\beta^+ - \beta^- \ll 10^{-2}$) while the search space of β can be arbitrarily large.

We denote this as the *Reconstruction-Concentration Dilemma* (RCD) and in this paper we investigate possible solutions that can increase the variety of items under the over-concentration case. To achieve this, one can simply apply post-generation perturbation to enforce item variety, yet this solution ignores the intra-slate features and significantly downgrades the recommendation accuracy. With this in mind, we further derive a modification of the original generation process, so that it can perturb before the final generation while reducing the negative effect of the perturbation. Specifically, when generating a slate, it follows a two-phase procedure: first, a pivot selection model chooses an item for a fixed slate position; then a slate completion model generates the remainder of the slate based on the pivot item along with other constraints. With this framework, we summarize our contributions as below:

- We propose to consider both the slate accuracy metric and the slate variation metric when evaluating models that generate stochastic slates.
- We identify the RCD with these metrics and show that the most desirable recommendation performance appears in a narrow "elbow" region.
- We conduct experiments on real-world datasets and simulation environments to show that enforcing variation can mitigate over-concentration and extend the elbow's performance to a wide range of search space.
- We show that the proposed pivot selection phase can provide better control over the slate variation under the over-concentration case of the dilemma.

In the following sections, we first list related studies in section 2, then describe how generative slate recommendation is achieved in section 3.1. Further, we explain how to employ variance metrics as complements of accuracy metrics in section 3.2, and then introduce our slate recommendation framework in section 3.3. We present our experimental results on both real-world datasets and simulation environments in section 4 and 5 as the evidence to support our claims. And finally, we discuss some other possible solutions that may also improve the item variety to bridge the gap between generative methods and recommendation systems.

2 RELATED WORK

There exist several types of generative modeling approaches to recommender systems. The most studied area is to leverage recurrent neural networks (RNN) [14]. It models the probability of each item conditioned on all previously recommended items $P(d_i|d_{i-1}, \ldots, d_1)$ and consecutively make recommendations from d_1 to d_K. Modeling in this way means that the recommendation of item d_i does not depend on the items $d_{i+1}, \ldots, d_K$ that appear later, which weakens the intra-list relation of the recommendations. This sub-optimality has already been shown in [28]. Another track of research uses auto-encoder for recommendation [32, 38], but they model the user history profiles instead of the distribution of slates. A recent line of research adopts reasoning-based recommendation models [11, 40, 47], which models recommendation as a cognition rather than perception task and adopts neural reasoning rather than neural matching models for better recommendation.

In addition to the generative approach represented by [28], there are other efforts that aim to deal with slate recommendation using reinforcement learning (RL) [16, 26, 27, 42]. Like the early attempts [39, 43], this type of methods mostly targets on exploring how to make use of the long term effects of several consecutive recommendations by transforming the slate and its user reaction as "states" in RL. Though they are suitable for solving the problem of slate recommendation, the essence behind RL and generative methods are mostly complementary, since a generated model can be pretrained and transplanted as the actor in RL frameworks.

We can also consider slate recommendation as a type of list recommendation, but the list size is fixed. Except for accuracy measures, there are many list-wise metrics that are proved beneficial to both the recommender systems and its customers, including but not limited to coverage [19] and intra-list diversity [49, 50]. Typically, the solution has to balance between accuracy and diversity, such as Max-Marginal Relevance (MMR) [9], relative benefits [6], α-NDCG [12], and Determinantal Point Process (DPP) [15]. But as pointed out by Jiang et al. [28], it will be unfair to compare these essentially discriminative methods in generative settings, and conversely, it will be unfair for generative methods to compete in traditional LTR settings. In order to show this deviation, we investigate how much discriminative ranking methods are different from generative methods if compared in the same setting in section 5.

A relatively unrelated track that considers slate-wise patterns is to re-rank the items based on the expected user interaction on the candidate slate [1, 3, 46]. However, the items available for re-ranking are often restricted to the candidates given by some base ranking model. Our problem is about directly generating slate recommendations with no restriction on candidate items, which is essentially a different task. One should also distinguish slate recommendation with session-based recommendation [22], which usually consists of user interaction history of arbitrary length, typically with a sequence of sessions, and the major research focus is on the modeling of the user sequential behaviors [14, 41].

3 GENERATIVE SLATE RECOMMENDATION

The corpus of items is denoted as $\mathcal{D}$, and a slate of size K is defined as an ordered list of items $\boldsymbol{s} = (d_1, d_2, \ldots, d_K)$, where $d_k \in \mathcal{D}$ and positional index $k \in \{1, \ldots, K\}$ represents that the item appeared in the k-th slot in the slate. A user's response to a slate $\boldsymbol{s}$ is denoted as $\boldsymbol{r} = (r_1, r_2, \ldots, r_K)$, where r_k is the response on item d_k, e.g., $r_k \in \{0, 1\}$ represents d_k is clicked or not. Assume that each slate $\boldsymbol{s}$ has corresponding latent unknown features $\boldsymbol{z}$ and some known characteristics/constraints $\boldsymbol{c}$. Typically, let $\boldsymbol{c} = \text{onehot}(\sum_{k=1}^{K} r_k)$ so that the user responses are contained in the constraints. For example, for a slate with 0 click, the corresponding constraint would be $[1, 0, 0, 0, 0, 0]$, while for a slate with 3 clicks, the constraint would be $[0, 0, 0, 1, 0, 0]$. Unlike discriminative ranking methods that model $R(\boldsymbol{r}|\boldsymbol{s})$, which is the user response for a given slate, the goal of generative slate recommendation models is to learn the distribution of slates with the given constraints:

$$P_\theta(\boldsymbol{s}|\boldsymbol{z}, \boldsymbol{c})$$

where z is the latent slate encoding. An optimal slate s^* should maximize the expected number of clicks $\mathbb{E}[\sum_{k=1}^{K} r_k]$, so during recommendation, one should provide to the inference model with the maximum number of clicks as constraint $c^* = [0, 0, 0, 0, 0, 1]$ (correspond to the ideal all-clicked response $r^* = [1, 1, 1, 1, 1]$). Different from the setting in [28], we also allow user features, so the constraint vector c in this case will be the concatenation of extracted user embedding and the aforementioned transformed response. As we will discuss in section 5.4, a more fine-grained constraint vector that involves user is more likely to induce a smooth distribution instead of a disjoint manifold in the encoding space z.

3.1 Slate Generation Model

To find a good generative model $P_\theta(s|z, c)$, a Conditional Variational Auto-Encoder (CVAE) framework learns a set of latent factors $z \in \mathbb{R}^m$ such that z can encode sufficient high-level information to reproduce the observed slates with maximum likelihood. As formulated in [30], a variational posterior $Q_\phi(z|s, c)$ is used as an approximation to solve the intractable marginal likelihood (which involves integral over latent z). The resulting model structure contains an encoder Q_ϕ that learns to encode the input slate s and constraint c into a set of variational information (e.g., the mean and variance when Gaussian prior is assumed) of each factor of z, and a decoder P_θ, which corresponds to the generative model. When training the model, one can maximize the variational Evidence Lower Bound (ELBO) of the data likelihood [30], which is equivalent to minimizing:

$$\mathcal{L}_s = \mathbb{E}_{Q_\phi(z|s,c)}\left[\log P_\theta(s|z, c)\right] - \beta \mathrm{KL}\left[Q_\phi(z|s, c)\|P_\theta(z|c)\right] \quad (1)$$

where $P_\theta(z|c)$ is the conditional prior distribution of z, KL represents the Kullback-Leibler Divergence (KLD), which restrains the distance measure between two distributions, and β is the trade-off coefficient as described in section 1. The encoder, decoder, and the conditional prior are all modeled by neural networks to capture complex feature interactions. With the decoder, items of each slate are selected based on the dot product similarity between output embeddings and embeddings of all items in $\mathcal{D}$. During training, in order to avoid overfitting, the reconstruction loss is calculated by the cross entropy over down-sampled items instead of the entire $\mathcal{D}$. At inference time, the slate is generated by passing the ideal condition c^* into the decoder along with a randomly sampled encoding z (e.g., from random Gaussian) based on the variational information provided by the conditional prior.

In the loss Eq. (1), we can interpret the KL divergence as how well the learned encoding z distribution is regulated by the guiding prior $P_\theta(z|c)$, and the other term reveals how well existing slates are reconstructed. According to [23], manipulating the trade-off parameter β will push the model to favor one of the terms over the other. For example, if we assume isotropic Gaussian as the prior distribution and set larger β, the factors in the learned z space will become more disentangled, and thus more meaningful control over the generation, but with a possible downgrade of reconstruction performance resulting in unrealistic generation. Despite its feasibility in many other tasks, as we will discuss in section 5.1, this β leads to a reconstruction-concentration trade-off that barely provide satisfactory recommendation results.

3.2 Variance Evaluation of Generated Slates

Many generative methods (e.g. VAEs and GANs[20]) are stochastic in terms of the output, but it is possible that the slate encoding z is not obtained through an encoder model so one cannot simply estimate the slate variance based on z. Thus, we are interested in evaluation metrics that can estimate the variance of slates for a wide range of stochastic models.

An evident choice is directly using **item variance** across all possible generated slates. Since items are typically represented by embedding vectors, let $x_1, \ldots, x_K$ be the vector representations of generated items. For simplicity, assume conditional independence among factors of x, then the item variance can be calculated as the variance of each factor and be approximated by sampling:

$$\begin{aligned}\mathrm{Cov}(x) &= \mathbb{E}_{s\sim P_\theta}\left[\frac{1}{K}\sum_{i=1}^{K}\left\|x_i^{(s)} - \mu\right\|^2\right] \\ &= \lim_{N\to\infty}\frac{1}{NK}\sum_{j=1}^{N}\sum_{i=1}^{K}\left\|x_i^{(s_j)} - \mu\right\|^2\end{aligned} \quad (2)$$

where N is the number of generated slate samples, and each slate s_j is sampled from $P_\theta(s|\cdot)$. Note that μ is the average of all NK generated items, and it depends on the input constraint. If the generative model is personalized, then the user is included in the input of P_θ and the generation process will first run N times for each user to give personalized variance estimation, then the estimations are averaged for all users.

Let $\mu^{(s)}$ be the average item of slate s:

$$\mu^{(s)} = \frac{1}{K}\sum_{i=1}^{K} x_i^{(s)} \quad (3)$$

then each slate variance in Eq.(2) can be decomposed into:

$$\begin{aligned}&\sum_{i=1}^{K}\|x_i^{(s_j)} - \mu\|^2 = \sum_{i=1}^{K}\|x_i^{(s_j)} - \mu^{(s_j)} + \mu^{(s_j)} - \mu\|^2 \\ &= \Bigg(\sum_{i=1}^{K}(x_i^{(s_j)} - \mu^{(s_j)})^\top(x_i^{(s_j)} - \mu^{(s_j)}) + \sum_{i=1}^{K}(\mu^{(s_j)} - \mu)^\top(\mu^{(s_j)} - \mu) \\ &+ 2(\mu^{(s_j)} - \mu)^\top\sum_{i=1}^{K}(x_i^{(s_j)} - \mu^{(s_j)})\Bigg)\end{aligned} \quad (4)$$

Since the last term has $\sum_{i=1}^{K}(x_i^{(s_j)} - \mu^{(s_j)}) = 0$ (from Eq.(3)), it simplifies the total item variance as:

$$\mathrm{Cov}(x) = \lim_{N\to\infty}\frac{1}{N}\sum_{j=1}^{N}\left\|\mu^{(s_j)} - \mu\right\|^2 + \frac{1}{NK}\sum_{j=1}^{N}\sum_{i=1}^{K}\left\|x_i^{(s_j)} - \mu^{(s_j)}\right\|^2 \quad (5)$$

where the first term describes the **slate-mean variance** and the second term describes the **intra-slate variance**. Each of the two terms provides a lower bound for the total item variance, and conversely, the total item variance Eq.(2) gives an upper bound for either term. A useful conclusion we can derive from this is that models good at one of the two terms in Eq.(5) may not be the one that achieves the best total item variance. On one hand, models with good intra-slate variance may still provide repeating slate with the same $\mu^{(s_j)} = \mu$, which results in extremely low slate-mean variance. On the other hand, models with good coverage of

item across slates may still have repeated items (in the most extreme case, $\boldsymbol{x}_i^{(s_j)} = \boldsymbol{\mu}^{(s_j)}$ when all items are equal) inside each slate inducing reduced intra-slate variance. Intuitively, we would like to make both slate-mean variance and intra-slate variance sufficiently large in order to support good total variance. Thus, the evaluation protocol should at least include two of the metrics among total item variance, slate-mean variance, and intra-slate variance.

3.3 The Two-Phase Generation Framework

We seek to enforce slate variation when CVAE model provides over-concentrated recommendations (i.e., the large β case of RCD). A straightforward solution is to perturb the generated slate by considering each position as a separate ranking model. However, this post-generation perturbation is very hard to control and always takes the risk of significant downgrade of recommendation accuracy (detail in Appendix A), due to the large perturbation space and the ignorance of the positional bias and item relations. With this in consideration, we turn to pre-generation perturbation and propose a simple and effective CVAE framework to mitigate the problem. In general, we separate the original generative process into two steps:

$$\begin{aligned} P_\theta(\boldsymbol{s}|\boldsymbol{z},\boldsymbol{c}) &= P_\theta(d_1,\ldots,d_K|\boldsymbol{z},\boldsymbol{c}) \\ &= P_\theta(d_2,\ldots,d_K|d_1,\boldsymbol{z},\boldsymbol{c})P_\theta(d_1|\boldsymbol{z},\boldsymbol{c}) \end{aligned} \tag{6}$$

That is, the framework first uses a *pivot selection model* $P_\theta(d_1|\boldsymbol{z},\boldsymbol{c})$ to select an adequate pivot item for a fixed slate position (here d_1 means we always generate the first appearing item in the slate). Then with this pivot item as additional condition, a *slate completion model* $P_\theta(d_2,\ldots,d_K|d_1,\boldsymbol{z},\boldsymbol{c})$ generates the rest of the items for the slate. With this separation, we can avoid RCD by enforcing variation of resulting slates through perturbation in the first stage, and use the second phase to clean up the mess if it has made a bad choice of pivot. As illustrated in Figure 1, the pivot controller is only applied to the generative decoder. Compared to the standard VAE model, little has to be nudged in the encoder $Q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{c})$ since it already has the potential to encode any intra-slate pattern.

Picking Pivot Item for the Slate: $P_\theta(d_1|\boldsymbol{z},\boldsymbol{c})$ will predict an item as the pivot, based on this, the slate completion model will fill in the rest of the slate according to the pivot. In other words, the goal of this part is to find the best item for a certain position in the slate, based on the encoding $\boldsymbol{z}$ and constraint $\boldsymbol{c}$. It first generates an "ideal" latent item embedding $\widehat{\boldsymbol{x}}_1$, and then applies dot product with all item embeddings in Ψ to find the closest item as the d_1. The minimization of the reconstruction term can be achieved by optimizing the cross entropy with softmax. In practice, we also use down sampling [35] to reduce the computational cost and alleviate over-fitting on the training set. Readers may notice that this part can be treated as a typical ranking model and thus any learning-to-rank framework is suitable for its training, only that one instead of many items are selected at a time.

Similar to sequential modeling, the training of $P_\theta(d_1|\boldsymbol{z},\boldsymbol{c})$ is made independent of the later slate completion model, and in both training and inference, this pivot selection phase allows perturbation which improves the item variation. Yet, perturbation inevitably causes information loss and downgrades the recommendation accuracy. Theoretically, taking the simplest assumption that item interactions are directional and are all binary relations, then there are at most $K(K-1)$ such interactions between items for a slate of size K. This separation and the introduction of perturbation mean that our model neglects $K-1$ of them (from $K-1$ remaining items towards the pivot). Even though, in our experiments, we find that this pre-generation perturbation can improve item variety more significantly with only a minor loss of accuracy compared to post-generation perturbations, which means that the later slate completion model is able to correct the slate according to the perturbed pivot. Additionally, we suggest to pick one pivot instead of more in this phase, since for any $1 < k' < K$ (in the binary relation case), when choosing k' pivots, the number of missing relations will be $(K-k')k' \geq K-1$, which indicates more loss of information and recommendation accuracy.

Slate Completion with a Given Pivot Item: After the selection of the pivot, the goal of the slate completion model

$$P_\theta(d_2,\ldots,d_K|d_1,\boldsymbol{z},\boldsymbol{c}) \tag{7}$$

is to learn to fill up the remaining items that can satisfy the desired constraint $\boldsymbol{c}$. A forward pass will take as input the selected pivot $\widehat{d}_1$, the encoding $\boldsymbol{z}$ (which is the output of Q if training, output of the conditional prior $P_\theta(\boldsymbol{z}|\boldsymbol{c}^*)$ if inference, as in VAE Eq.(1)), and the constraint $\boldsymbol{c}$, then output a set of "best" latent item embeddings $\widehat{\boldsymbol{x}}_2,\ldots,\widehat{\boldsymbol{x}}_K$ for each of the remaining slots in the slate. After generating these latent embeddings, it will find for each of the $\widehat{\boldsymbol{x}}_i$ the nearest neighbor in the candidate set $\mathcal{D}$ through dot product similarity. Similar to that in the pivot selection model, we can again apply cross-entropy loss with softmax and negative sampling during training. Note that this is the final generation stage and it does not employ perturbation.

Compared to inference time when the model can only use the inferred $\widehat{d}_1 \sim p_\theta(d_1|\boldsymbol{z},\boldsymbol{c})$ from the pivot selection model, during training, there is another valid choice of the pivot - the ground truth item in the data. We find that the later choice achieves the same performance but usually exhibits faster convergence. Thus, we adopts the ground truth item d_1 for the input of the slate completion model during training in our experiments, and if perturbation, we calculate item similarities based on the ground truth instead of the inferred item embedding. Additionally, when the pivot is perturbed during training, the slate completion model tends to learn a "denoised" intra-slate patterns which may results in a slate that is more accurate but with less variation, compared to training without perturbation, as we will discuss in section 5.3.

4 EXPERIMENTAL SETTING

4.1 Real-world Datasets

We conducted experiments [1] on two real-world datasets. The first is **YOOCHOOSE** [2] from RecSys 2015 Challenge and we follow the same reprocessing procedure as [28]. The resulting dataset contains around 274K user slate-response pairs. Note that there is no user identifier involved in this dataset, so our second dataset is constructed from the MovieLens 100K[3] dataset. We split user rating sessions into slates of size 5 and consider the rating of 4-5 as positive feedback (with label 1) and 1-3 as negative feedback

[1] Code link: https://github.com/CharlieFaceButt/PivotCVAE
[2] https://2015.recsyschallenge.com/challenge.html
[3] https://grouplens.org/datasets/movielens/100k/

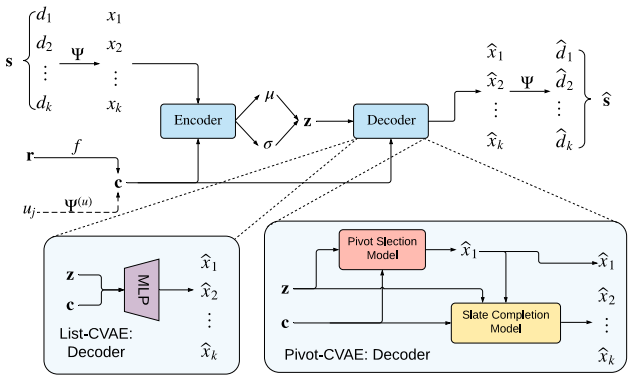

Figure 1: Structure of the generative framework during training. s is the input slate of size k. r is the user response vector of the input slate. $\hat{s}$ represents the output slate inferred by decoder. Ψ and $\Psi^{(u)}$ extract pretrained embeddings for items and users, respectively.

(with label 0). The resulting distribution of slate responses (Figure 8 in Appendix C) is similar to that in the Yoochoose dataset. We consider two versions of this dataset: **ML (User)** and **ML (No User)** to investigate how the presence of user affects the generative results. Compared to ML(User), the ML(No User) dataset ignores user IDs like Yoochoose data. Since both datasets are skewed towards slates with 0 and 3 clicks, we augment the records of 1,2,4, and 5 clicks by random repetition until each group has at least half the size of the largest response type. Note that these offline log data have limited feasibility for evaluation since they cannot provide accurate estimations for unseen records. Thus, an additional user response model $R : \mathcal{D}^K \rightarrow \{0,1\}^K$ is trained (with binary cross-entropy loss) to fulfill the role of "ground truth" user feedback.

4.2 Simulation Environment Settings

To observe how generative models behave for unseen slates under different environment settings and to investigate the difference between slate generation metrics and traditional ranking metrics, similar to existing works [26, 28], we employ simulations with plugins of positional biases and item interactions.

The primary goal of the simulated environment is to model $R(\boldsymbol{r}|\boldsymbol{s},\boldsymbol{u})$ that predicts the users' true responses given slate $\boldsymbol{s}$. And for each of the simulators described in this section, the final response for each item d_k is sampled by Bernoulli distribution with click probability $\mathcal{I}(d_k, j)$, which represents user j's interest for d_k:

$$r_k = R(r_{kj}|d_k, j) \sim \text{Bernoulli}(\mathcal{I}(d_k, j)) \tag{8}$$

Thus, the click behavior follows Poisson binomial distribution, and the expectation of the number of clicks is:

$$\mathbb{E}\Big[\sum_{k=1}^{K} r_k\Big] = \sum_{d_k \in \boldsymbol{s}} \mathcal{I}(d_k, j) \tag{9}$$

We tune the resulting distribution with proper setting (details in appendix D) so that it coincides with that of real-world datasets.

Specifically, each simulation is built based on a basic **User Response Model (URM)**, which only considers point-wise user-item responses like the matrix factorization model. By adding awareness of positional bias and multi-item relations, we obtain **URM_P** (P stands for positional bias) and **URM_P_MR** (MR stands for multi-item relations), respectively. The URM_P_MR consists of a coefficient ρ for the weight of the multi-item relations. As a special case, setting $\rho = 0$ will tell the simulation to include no item relations and the environment will reduce to **URM_P**. The details of each simulation environment are given in Appendix D.

Table 1: Pivot-CVAE variations

Models	perturbation of d_1	
	training time	inference time
Pivot-CVAE (GT-PI)		
Pivot-CVAE (SGT-PI)	✓	
Pivot-CVAE (GT-SPI)		✓
Pivot-CVAE (SGT-SPI)	✓	✓

Simulation Data: We set up three URM_P_MR environments ($|\mathcal{D}| = 3,000$, $|\mathcal{U}| = 1,000$) with different values of $\rho \in \{0, 0.5, 5.0\}$. Note that there is no need to train a response model from the generated dataset like that for real-world datasets. Conversely, we generate a training set of 100,000 slates from each environment. The number of slates for all types of user responses are also balanced similar to that of real-world datasets. The user and item embeddings are assumed explicit and free to use in the training of the recommendation model. Here we expect readers to distinguish these simulations from those used in Reinforcement Learning (RL)-based recommendation models, because the generative model does not interact with the simulated environment for rewards during training. In other words, the generative model is training offline and the simulators are only used for evaluation purposes.

4.3 Model Specification

We denote our two-step generative process as Pivot-CVAE. For Pivot-CVAE model, perturbation of d_1 can be applied either on training phase or inference phase, inducing 4 possible variations: where "GT" represents that the model uses Ground Truth item during training, "PI" represents that the model uses Pivot Item during inference, and "S" means the item applies perturbation. For all perturbation, we adopt sigmoid dot-product between item embeddings as similarity and sample according to multinomial distribution so that it can capture user interests.

Baselines: We include the **List-CVAE** model [28] as an example of VAE and build its non-greedy version (denote as **Non-Greedy List-CVAE**) that conducts post-generation perturbation. That is, after the generation of the slate, the item d_1 (in the same position as the pivot of Pivot-CVAE) is perturbed by sampling from $\mathcal{D}$. Again, we apply sampling based on multinomial distribution of sigmoid dot product similarity. We also include biased **MF** [31] and **NeuMF** [21] as representatives of discriminative ranking models. In order to engage generative recommendations that can explore items other than the top items, we extend these discriminative methods into **Non-greedy MF/NeuMF** by applying the same perturbation method on d_1 as that in Non-greedy List-CVAE and Pivot-CVAE. To compare the item variance with intra-slate variance, we include the widely adopted **MF-MMR** [10] as a representative diversity-aware method. It re-ranks the items proposed by the pre-trained biased

MF model based on the following modified score:

$$\text{score}(d) = \lambda \,\text{sim}(d, j) + (1 - \lambda) \max_{d_i \in \boldsymbol{s}} \text{sim}(d_i, d)$$

where slate $\boldsymbol{s}$ starts from an empty set and choose the item with the best MMR score in each step until the slate size is K. $\text{sim}(d, j)$ represents the item's original ranking score given by the base MF model, and $\text{sim}(d_i, d)$ is the item's similarity to the i-th item that has already been added to the list $\boldsymbol{s}$.

In our experiment, we adopt two-layered network with 256 dimensional hidden size for each encoder, decoder, $P_\theta(d_1|\boldsymbol{z}, \boldsymbol{c})$, the slate completion model $P_\theta(d_2, \ldots, d_K|d_1, \boldsymbol{z}, \boldsymbol{c})$, and the MLP component in NeuMF. In terms of the performance of CVAE-based models, we found that it is relatively insignificant to change the width or depth of the encoder and decoder network as long as they are large enough. The user and item embedding size for all datasets and simulations are fixed to 8, and the size of $\boldsymbol{z}$ is set to $m = 16$. The slate size is $K = 5$, which means the size of the condition input $\boldsymbol{c}$ of CVAE-based model is $K + 1 = 6$ (without user condition) as described in the first paragraph of section 3. All models are optimized by Adam with stochastic mini-batches (batch size of 64), and we use grid search to find the best learning rate (0.0001 for List-CVAE and Pivot-CVAE, 0.0003 for MF and NeuMF) and weight decay (0.0001 for all models). For MF and NeuMF, we follow the standard LTR paradigm with point-wise binary cross-entropy loss and assign 2 random negative samples of each record to optimize these models until their ranking performance converges in the validation set. For MF-MMR, we use sigmoid dot-product as item similarity and set $\lambda = 0.5$. During training of generative models, the softmax function on each slot in a slate is associated with 1000 negative samples for Yoochoose, and 100 negative samples for MovieLens and simulation environments.

4.4 Evaluation Protocol

For all datasets, we randomly split them into train, validation, and test sets following the 80-10-10 holdout rule. And we run each experiment five times to obtain the average performances. We consider two major evaluation metrics based on interactive environment $R(\boldsymbol{r}|\boldsymbol{s})$: slate accuracy and slate variation. And for the illustration of why ranking metrics on test set is invalid for evaluation of generative models, we further include discriminative ranking metrics.

Slate Accuracy Metric: The primary metric, following the evaluation setting of [28], is the *Expected Number of Clicks* (ENC) which is calculated as:

$$\mathbb{E}\left[\sum_{k=1}^{K} r_k\right] = \sum_{\boldsymbol{s} \in \mathcal{D}^K} P(\boldsymbol{s}) \mathbb{E}\left[\sum_{k=1}^{K} r_k | \boldsymbol{s}\right]$$

where $\boldsymbol{r}_k|\boldsymbol{s}$ is a random variable modeled by $R(\boldsymbol{r}|\boldsymbol{s})$, and $P(\boldsymbol{s})$ is the probability of generating $\boldsymbol{s}$. Similar to the variation evaluation described in section 3.2, we can approximated this metric by sampling techniques. This metric is exactly the ultimate goal of the optimization and does not involve any test set compared to traditional ranking metrics. For simulation, combining Eq. (9), it becomes:

$$\mathbb{E}\left[\sum_{k=1}^{K} r_k\right] = \sum_{\boldsymbol{s} \in \mathcal{D}^K} P(\boldsymbol{s}) \sum_{d_k \in \boldsymbol{s}} \mathcal{I}(d_k, j)$$

And for real-world dataset, we train $R(\boldsymbol{r}|\boldsymbol{s})$ ($R(\boldsymbol{r}|\boldsymbol{s}, u)$ if user IDs are presented) with point-wise binary cross entropy minimization.

Slate Variation: This metric reveals the severance of the "over concentration" in RCD and the generation pitfall of limited slate prototypes. As described in section 3.2, we use total item variance and intra-slate variance metrics in our evaluation. Notably, the variance of $\boldsymbol{z}$ directly models the slate variance, but it is unique in VAE-based generative models. In order to form comparison with non-VAE models, we use item *Coverage* [19] as the item variance metric and *Intra-List Diversity* (ILD) [49, 50] as an approximation of the intra-slate variance. Item coverage estimates the proportion of unique items in $\mathcal{D}$ that can appear after several times of generations. Obviously, LTR models are deterministic so will always cover only $5/|\mathcal{D}|$ of the items without perturbation. Intra-list diversity is based on Intra-List Similarity (ILS) [50]:

$$\text{ILD} = 1 - \text{ILS}(\boldsymbol{s}) = 1 - \sum_{d_i \in \boldsymbol{s}} \sum_{\substack{d_l \in \boldsymbol{s} \\ d_l \neq d_i}} g(\boldsymbol{v}_i^\top \boldsymbol{v}_l)$$

where the similarity measure g between d_i and d_l in the slate is based on the dot product of their item embeddings.

Ranking Metrics: We agree with [28] that it is inadequate to use traditional offline ranking metrics to evaluate generative models, as we will discuss in section D.1, these metrics behave differently on a test set compared to that on a interactive user response environment. Even though, it is still reasonable to compare these metrics among generative models. Specifically, we calculate slate *Hit Rate* and slate *Recall* considering each slate as a ranking list. It is considered as a "hit" if an item in the ground truth slate with positive feedback is recommended. And the slate recall considers each slate as a user history instead of the combined user history across slates. Note that in Yoochoose and ML, user identifiers are absent, so we assume a universal user for all slates during training.

In summary, we conduct two types of evaluation: 1) recommendation performance (slate accuracy and variance metric) on $R(r|s)$ as main evaluation, and 2) ranking metric on the test set. Due to the stochastic nature of generative models (List-CVAE, Pivot-CVAE, and all Non-greedy models), the evaluation of each metric is calculated based on N sampled outputs (correspond to section 3.2). Note that N cannot be too small or else it will not provide accurate and stable estimation of the true value. In the meantime, it can neither be too large, otherwise the model would exhibit indistinguishably high item coverage (i.e. it may simply generate all items in $\mathcal{D}$ given sufficient number of samples).

5 RESULTS AND DISCUSSIONS

5.1 The Reconstruction-Concentration Dilemma

We consider the search space of $\beta \in [0.00001, 30.0]$ (chosen uniformly on $\log \beta$ space) and for each setting we train List-CVAE and all Pivot-CVAE models until convergence of ENC on $R(\boldsymbol{r}|\boldsymbol{s})$. When evaluation, we generate $N = 500$ slates from each trained model and calculate the average as described in section 4.4. In Figure 3, we plot the RCD pattern of List-CVAE on Yoochoose dataset, and we have observed the same pattern in MovieLens 100K and all simulation environments.

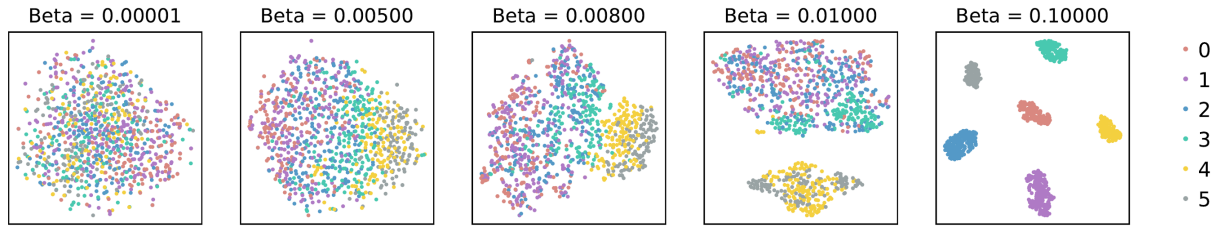

Figure 2: The slate encoding TSNE plots of List-CVAE on Yoochoose. The first plot correspond to over-reconstruction case, the last corresponds to over concentration case, and the middle plots correspond to the "elbow" case.

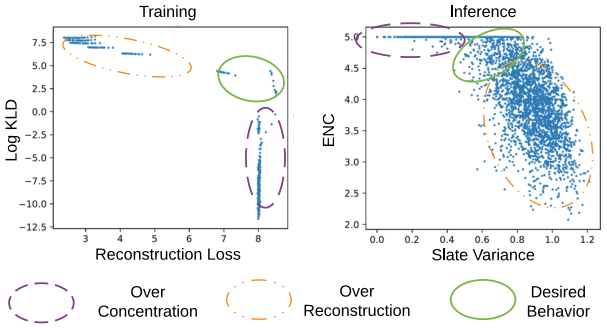

Figure 3: Training loss behavior (left) and recommendation performance (right) of RCD on the Yoochoose Data. Each point in the left panel represents the average result of slates in one training epoch of a model. Each point in the right panel represents a certain generated slate. Here, we use ILD as slate variation, ENC as accuracy metric.

In cases where β is small, CVAE becomes biased towards learning the reconstruction term of Eq. (1) as illustrated by the yellow dot-dashed circle in the left subplot of Figure 3. And because of the subdued regularization from the KL term, the encoding distribution of $\boldsymbol{z}$ becomes less aligned with that of the predefined prior. When setting the prior $P_\theta(\boldsymbol{z}|\boldsymbol{c})$ as isotropic standard Gaussian, we observe that the means of the inferred $\boldsymbol{z}$ are often significantly deviated from $\mathbf{0}$ and the variances $\text{var}(\boldsymbol{z})$ are far from $\mathbf{1}$. Though it successfully learns and generates the slates in the dataset during training, there is no guarantee on the effectiveness of the sampled $\boldsymbol{z}$ during inference. In other words, the distribution of generated slates is close to a random selection on the observed dataset. As shown in the yellow dot-dashed circle in the right subplot of Figure 3, we observe low ENC and high variance during inference.

On the contrary, in the over-concentration case where β is rather large, the KL term plays a more important role in the learning. The slate encoding $\boldsymbol{z}$ indeed is more aligned with the prior, ensuring the sampling effectiveness, and consequently able to generate satisfactory slates during inference. Yet, it is less capable of encoding information that is necessary to reconstruct the slates. When the model learns that $\boldsymbol{z}$ is reluctant to encode corresponding slates, the generator tends to ignore $\boldsymbol{z}$ and focuses on the condition $\boldsymbol{c}$. Since $\boldsymbol{c}$ alone does not contain any variational information about slates, the model will only be able to output several biased "slate prototypes" (as illustrated in Appendix B, second row of Figure 6). An alternative analysis of the slate encoding $\boldsymbol{z}$ of List-CVAE is given in Figure 2. It shows that with large β, slate encoding becomes disjoint according to the ground truth number of clicks, which means that slate encoding tends to gather around its corresponding prototype given by the prior. This is undesirable since the model cannot infer slates outside the cluster, which results in the lack of variety in recommendations.

Besides, we notice that in the training data a lot of repeated clicks appear in the click and/or purchase sessions in Yoochoose data. This makes the RCD problem even worse since the same item is repeatedly recommended even within the same slate, inducing low intra-slate variance. We observed that RCD exists even with β-annealing [5], disabled condition (reduce CVAE to VAE), and constrained variation (only fix the variation of z, but not the mean). These observations indicates that RCD problem may exist for a broad range of generative models.

5.2 The Narrow "Elbow" of CVAE

Though neither of the extremes appears to be a good choice for recommendation, we find that there exists a very narrow region in between, where models can provide feasible outputs. In Figure 4, we show case the results of all metrics on ML(No User) data for generative models across different $\beta \in [0.00001, 10.0]$. X-axis represents the setting of β and note that results for different βs correspond to different models that are separately trained and evaluated. For ENC and ILD metrics, we use box plot to better demonstrate the distribution of generated slates.

We summarize three trends of model behavior when increasing the value of β as follows:

- For model training, the converged reconstruction loss gets worse while the KLD loss gets better;
- When inference, the accuracy measure ENC starts to boost but the variation metric of the generated slates drops;
- $\boldsymbol{z}$ starts to show clustering behavior under the regulating prior and the clusters will become crisper along with the transition as shown in Figure 5.

This transitional behavior indicates that models in this intermediate region can to some extent cover the variety of slates in the data while provide moderate accuracy performance. To better show the detailed transitional behavior of the feasible region, we include a more fine-grained search space for $\beta \in [0.001, 0.01]$ and highlight it with shaded green in Figure 4. However, in the experiment of both real-world datasets and all simulation datasets, we found that this transition happens within a very small region (at most 30% of the $\log \beta$ search space or equivalently 2% of the β search space), while the search space in our experiment is $\beta \in [0.00001, 30.0]$

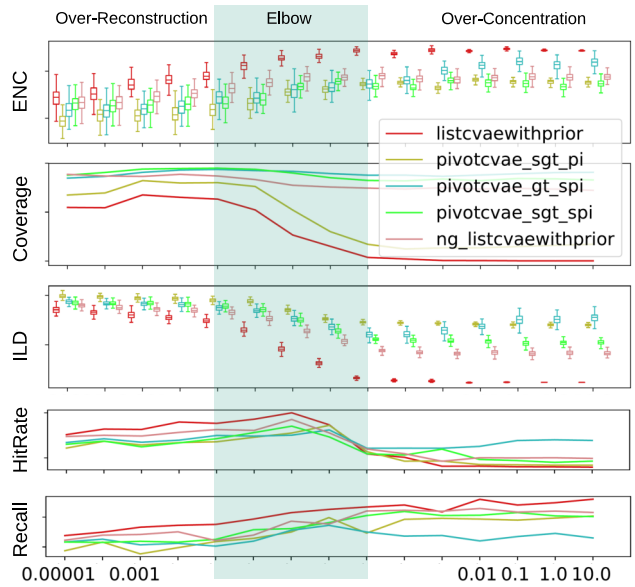

Figure 4: Performance on ML (No User) data. "listcvaewithprior" represents the List-CVAE and "ng_listcvaewithprior" corresponds to the Non-Greedy List-CVAE.

Additionally, we observe that this transitional region consistently gives good test set ranking performance (both hit rate and recall) compared to other choices of β. The two extreme cases outside the "elbow" region do not always reveal a decreasing hit rate and an increasing recall on the test set as in Figure 4, but the best ranking performance usually appears in one of the two sides. Intuitively, the generative model should be able to maximize the likelihood of the test set in addition to the training set. Following the same derivation of Eq.(1), this would require both the ability to reconstruct the slate information and the ability to satisfy the constraint. This can only be observed in this transitional region if the slate variation is not enforced, since the two extremes only possess one of the two characteristics. Note that this ranking performance can only serve as indicators to compare generative models, and it is incomparable between deterministic ranking models and stochastic generative models. As we will discuss in section D.1, the stochastic generation process explores and proposes various good slates in the view of the user $R(\boldsymbol{s}|\boldsymbol{c})$, and may not necessarily pin-point the data in the test set thus it is typically not favored by this kind of metrics.

5.3 Controlling Slate Variation

We present the results of ENC and variance in Table 2. Generative models with $\beta = 1.0$ are chosen as representatives of the large-β case, since we want to observe the improvement of slate variance when models are over-concentrated. Generative models with small β (described in section 5.1) and post-perturbation methods that change more than one item cannot provide satisfactory user response, so they are not included in the comparison. We only present results of datasets with user IDs (ML (User) and all simulation environments) so that collaborative filtering models like MF and NeuMF can be compared. The result of each stochastic model (Non-Greedy models, List-CVAE and Pivot-CVAE models) is calculated by the

Table 2: Model Performance on User Feedback $R(r|s)$ of datasets with user IDs. All results are significant ($p < 0.05$) and the overall best are the bold scores while the best among generative models are underlined.

	ML(User)	URM_P	URM_P_MR (ρ=0.5)	URM_P_MR (ρ=5.0)
A: Expected Number of Click (ENC)				
MF	3.246	**3.353**	3.870	4.961
NeuMF	3.197	3.344	3.810	4.938
MF-MMR	2.400	3.243	3.725	4.617
Non-Greedy MF	2.950	3.315	3.755	4.869
Non-Greedy NeuMF	3.020	3.303	3.730	4.819
List-CVAE	**3.579**	3.237	3.924	**4.971**
Non-Greedy List-CVAE	3.285	3.262	3.883	4.777
Pivot-CVAE (SGT-PI)	3.376	3.274	**3.934**	4.944
Pivot-CVAE (GT-SPI)	3.252	3.226	3.711	4.622
Pivot-CVAE (SGT-SPI)	3.152	3.270	3.816	4.704
B: Item Coverage				
MF	0.003	0.002	0.002	0.002
NeuMF	0.003	0.002	0.002	0.002
MF-MMR	0.003	0.002	0.002	0.002
Non-Greedy MF	0.142	0.082	0.082	0.080
Non-Greedy NeuMF	0.141	0.082	0.081	0.080
List-CVAE	0.004	0.030	0.011	0.005
Non-Greedy List-CVAE	0.139	0.106	0.088	0.084
Pivot-CVAE (SGT-PI)	0.071	0.065	0.014	0.005
Pivot-CVAE (GT-SPI)	**0.250**	**0.235**	**0.180**	**0.227**
Pivot-CVAE (SGT-SPI)	0.144	0.097	0.090	0.083
C: Intra-List Diversity (ILD)				
MF	0.206	0.031	0.035	0.036
NeuMF	0.694	0.300	0.534	0.779
MF-MMR	0.287	0.230	0.193	0.227
Non-Greedy MF	0.545	0.515	0.231	0.126
Non-Greedy NeuMF	**0.836**	0.576	0.644	**0.827**
List-CVAE	0.178	0.836	0.407	0.524
Non-Greedy List-CVAE	0.428	0.864	0.572	0.664
Pivot-CVAE (SGT-PI)	0.486	0.869	0.451	0.632
Pivot-CVAE (GT-SPI)	0.725	**0.945**	**0.740**	0.814
Pivot-CVAE (SGT-SPI)	0.551	0.856	0.600	0.637

average of all users' evaluation. Note that when calculating item coverage and diversity, we consider user-wise instead of the system-wise metric for these datasets.

The List-CVAE baseline achieves the best ENC on ML(User) and URM_P_MR environments because it is over-concentrated on the optimal slate prototype, and CF models achieves the best ENC on URM_P because of the pointwise environment. All models with item perturbation (Non-greedy List-CVAE, Pivot-CVAE (SGT_PI), Pivot-CVAE (GT_SPI), and Pivot-CVAE (SGT_SPI)) exhibit degraded ENC compared with the original List-CVAE, but significantly improves slate variation (Item Coverage and ILD). Among models using perturbation, the Pivot-CVAE (GT-SPI) model always achieves satisfactory accuracy with the best slate variety. We observe this outstanding performance across all datasets, meaning that sampling the pivot during inference (SPI) will induce more variance and explore more choices of item combinations than sampling

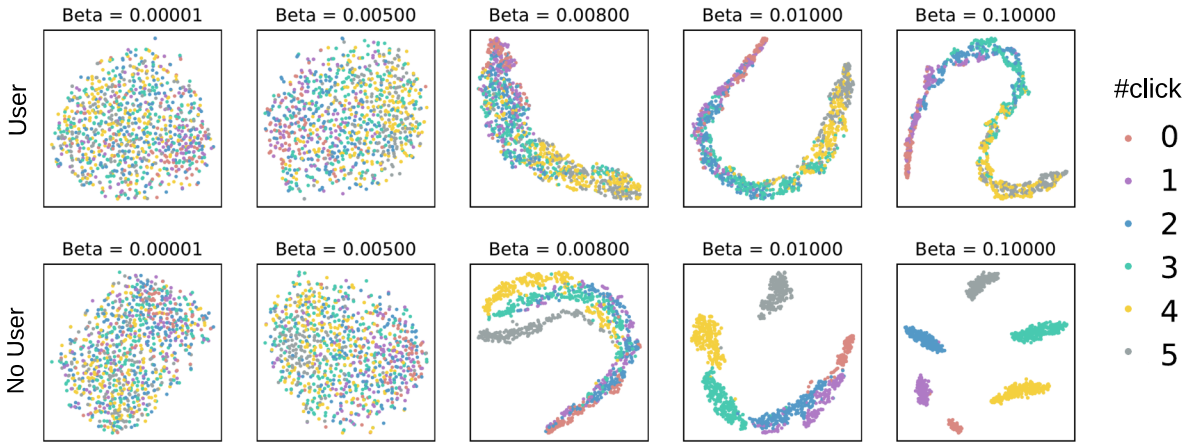

Figure 5: The slate encoding TSNE plots of List-CVAE on MovieLens datasets. When user identifier is presented, the encoding forms more fine-grained clusters that is no longer disjoint between one another.

during training (SGT). Pivot-CVAE (SGT_PI) applies perturbation during training but not inference, this allows the model to give more accurate generation with better ENC, but the improvement of item coverage and ILD becomes limited. Note that it can achieve a similar ILD with Non-Greedy List-CVAE even if there is a huge gap between their item coverages, indicating that SGT_PI seeks to find good slates with sufficient intra-slate variance but tends to be concentrated slate-wise in exchange for good accuracy. When applying perturbation on both training and inference as Pivot-CVAE (SGT-SPI), it has similar performance to Non-Greedy List-CVAE.

As shown in Table 2, generative methods consistently outperform MF and NeuMF on variance metrics, and achieves better ENC on all datasets except for URM_P where the environment is pointwise. This indicates that the user responses of real-world datasets like ML(User) are closer to URM_P_MR, which contain intra-slate features such as item relations, rather than URM_P. Additionally, Non-Greedy MF/NeuMF can improve the item coverage of these LTR models to the level of Non-Greedy List-CVAE baseline (still worse than Pivot-CVAE (GT-SPI)) and Non-Greedy NeuMF even occasionally achieves better ILD performance than Pivot-CVAE (GT_SPI). However, they achieve this with greater sacrifice on the ENC. On the other hand, MF-MMR is able to increase ILD, but its performance is worse than generative models on all metrics. Moreover, it also shows that a model that improves intra-slate variance does not necessarily improve the total item variance.

5.4 Personalization Improves Variance

Different from Yoochoose and MovieLens (No User) Data, the MovieLens (User) and our simulation environments include user ID in the constraints in addition to the ideal response, allowing the model to learn personalized preference of slates. We plot the distribution of z (of List-CVAE) in Figure 5 to show their difference in over-concentration case. For generative models trained with large β, instead of having disjoint slate encoding clusters for each type of user response, the presence of user ID in the constraint will guide the model to learn a set of more fine-grained clusters, each of which corresponds to a user. Note that the same user may have different types of user responses, and a typical user that usually gives a certain type of response also has a higher chance of giving responses of similar types (e.g., a user frequently clicks everything may also frequently click $K-1$ items). Consequently, user clusters become closer if they give similar types of response and closer response types become partially mixed with each other because of the common users, thus forming a topologically sorted chain in the space, as shown in the right most panels in the first row of Figure 5. This property will contribute to the total item variation of the overall system across users, but in a personalized view, the concentration of slate still exists. As given in Table 2-A, the user-wise item coverage of List-CVAE is close to that of discriminative ranking models.

6 CONCLUSION AND EXPECTATIONS

In this paper, we show that generative models for slate recommendation tasks may fall into the Reconstruction-Concentration Dilemma (RCD), where only a narrow middle region can produce effective recommendations. We point out that personalization or applying perturbation can enforce variation on the over-concentration case of the dilemma but have limitations. By separating a pivot selection phase from the generation process, we propose Pivot-CVAE mode that offers better control of the slate variation by perturbation before the generation. Our pivot-based approach and the variation evaluation framework can be extended to a wider scope of stochastic generation models such as Generative-Adversarial Networks (GAN) [45], which we will explore in the future. Besides, we also find it useful to construct a flexible and comprehensive user-response simulation framework, not only for the purpose of recovering a realistic recommendation environment but also for the need of generating unseen user-item interactions for model training and evaluation, which is essential for generative models as well as causal [4, 24, 34] and RL-based models. We will extend our framework for training and evaluating these models in the future.

ACKNOWLEDGMENTS

We thank Ji Zhang, Qi Dong, and all the reviewers for the constructive discussion.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *SIGIR*. ACM, 135–144.

[2] Qingyao Ai, Xuanhui Wang, Nadav Golbandi, Mike Bendersky, and Marc Najork. 2019. Learning Groupwise Scoring Functions Using Deep Neural Networks. In *Proceedings of the First International Workshop On Deep Matching In Practical Applications*.

[3] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[4] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM.

[5] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).

[6] Keith Bradley and Barry Smyth. 2001. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*. Citeseer, 85–94.

[7] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*. 89–96.

[8] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th ICML*. ACM, 129–136.

[9] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[10] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[11] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *Proceedings of the 30th Web Conference (WWW)*.

[12] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*. 659–666.

[13] Puneet Kumar Dokania, Aseem Behl, CV Jawahar, and M Pawan Kumar. 2014. Learning to rank using high-order information. In *European Conference on Computer Vision*. Springer, 609–623.

[14] 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*. ACM, 152–160.

[15] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. 2016. Bayesian low-rank determinantal point processes. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 349–356.

[16] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards Long-term Fairness in Recommendation. *arXiv preprint arXiv:2101.03584* (2021).

[17] Yingqiang Ge, Shuyuan Xu, Shuchang Liu, Zuohui Fu, Fei Sun, and Yongfeng Zhang. 2020. Learning Personalized Risk Preferences for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 409–418.

[18] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q. Zhu. 2019. Exact-K Recommendation via Maximal Clique Optimization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*.

[19] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. 1999. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence* (Orlando, Florida, USA). American Association for Artificial Intelligence, Menlo Park, CA, USA, 439–446.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th WWW*.

[22] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *4th International Conference on Learning Representations*.

[23] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Proceedings of 5th International Conference on Learning Representations*.

[24] Paul W Holland. 1986. Statistics and causal inference. *Journal of the American statistical Association* 81, 396 (1986), 945–960.

[25] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: A Configurable Simulation Platform for Recommender Systems. *arXiv preprint arXiv:1909.04847* (2019).

[26] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In *Proceedings of the Twenty-eighth IJCAI*. Macau, China, 2592–2599.

[27] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Navrekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. 2019. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767* (2019).

[28] Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy Mann, and Danilo J. Rezende. 2019. Beyond Greedy Ranking: Slate Optimization via List-CVAE. In *ICLR*.

[29] Thorsten Joachims, Laura A Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Sigir*, Vol. 5. 154–161.

[30] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[31] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[32] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.

[33] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[34] James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. 2020. Counterfactual Evaluation of Slate Recommendations with Sequential Reward Interactions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1779–1788.

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. [n.d.]. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*.

[36] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. 2009. Global ranking using continuous conditional random fields. In *Advances in neural information processing systems*. 1281–1288.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[38] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[39] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005).

[40] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural Logic Reasoning. In *CIKM*. 1365–1374.

[41] 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. *arXiv preprint arXiv:1904.06690* (2019).

[42] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-policy evaluation for slate recommendation. In *Advances in NIPS*. 3632–3642.

[43] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. 2007. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM RecSys*. ACM, 113–120.

[44] Paolo Viappiani and Craig Boutilier. 2010. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in neural information processing systems*. 2352–2360.

[45] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*. 515–524.

[46] Jianxiong Wei, Anxiang Zeng, Yueqiu Wu, Peng Guo, Qingsong Hua, and Qingpeng Cai. 2020. Generator and Critic: A Deep Reinforcement Learning Approach for Slate Re-ranking in E-commerce. *arXiv preprint arXiv:2005.12206* (2020).

[47] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *CIKM*. 1645–1654.

[48] Yisong Yue, Rajan Patel, and Hein Roehrig. 2010. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *WWW*. 1011–1018.

[49] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4511–4515.

[50] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on WWW*. ACM, 22–32.

APPENDIX

A POST-GENERATION PERTURBATION

Given a slate (of size 5) and its observed labels (#click) in data, we choose a numbers of its items randomly and apply perturbation to observe how much the ground-truth user response distribution $R(r|\boldsymbol{s})$ is affected. We present the result of the entire Yoochoose data (including val and test) in Figure 7. Each row corresponds to slates with certain observed label (#click), and each column corresponds to the number of perturbed item a. In each subplot, the x-axis corresponds to the ground truth expected user response $R(r|\boldsymbol{s})$ (from pre-trained model if Yoochoose/MovieLens dataset, and from simulation if URM-based environments), and the y-axis corresponds to the frequency/density of slates. As shown in the first column where no perturbation is involved, $R(r|\boldsymbol{s})$ is usually very close to the observed label. However, as given by the second column from left, changing merely a single item has already caused a significant deviation of distribution from that of the original slates, especially for slates with ideal condition $\boldsymbol{r}^*$ (the bottom row). And perturbation of 3 items is already inducing a distribution close to that of random slates (the rightmost column). We also observed this substantial reduction in MovieLens and simulation data. As shown in table 2, simply applying post-generation perturbation on a given slate without considering the context of the slate neither achieves the best ENC nor the best variation.

B MORE ON RCD

The detailed view of the reconstruction behavior of List-CVAE on the entire Yoochoose data is given by Figure 6. The same pattern also appears on MovieLens 100K. Each subplot gives the result of a List-CVAE model with certain β, the y-axis represents the predicted user response $R(\boldsymbol{r}|s)$ of the original slate and the x-axis represents $R(\boldsymbol{r}|\widehat{s})$ of the reconstructed slates. The reconstruction behavior on the dataset reveals its performance of inferring the observed dataset (including test set), which also helps identifying the RCD. Slates with the same observed #click have the same color, so ideally the slates with the same color should be somewhere close to the location [#click, #click]. We observe that the over-reconstruction happens when $\beta \leq 0.003$ (first three subplots), and the more distinguishable diagonal line in the plot with $\beta = 0.0003$ indicates a more severe reconstruction overfitting. The over-concentration happens when $\beta > 0.015$, and the reconstructed slates are typically gathered to the very point where the prototype slates are. This means that the decoder always tends to generate the same slate based only on the given constraint. Finally, the "elbow" transition between these two extreme cases happens in a narrow region around 0.01, where we observe several "prototypes" (the clustered vertical lines of each color) in each response group. In this region, the model shows moderate concentration behavior, but the resulting slates may still cover some degree of variance.

C SLATE RESPONSE DISTRIBUTION

The resulting slate response distribution of Yoochoose data and MovieLens 100K data are similar, and we show the later in Figure 8. Y-axis represents the frequency and X-axis correspond to the ground truth response $\boldsymbol{r}$ of slates. The label of X-axis is obtained by considering each $\boldsymbol{r}$ as a binary number and expressing it as integer. For example, user response $[0, 0, 0, 0, 1] \rightarrow 1_2 \rightarrow 1_{10}$ and $[1, 1, 1, 1, 0] \rightarrow 11110_2 \rightarrow 30_{10}$ where the subscript 2 and 10 means binary and decimal representation of numbers.

D DESIGN OF SIMULATION

Basic User Response Model (URM): We assume that the basic interaction between users and items follow a user interest model, which is a biased matrix factorization model[31]. Each item $d^i \in \mathcal{D}$ is associated with a vector $\boldsymbol{v}_i \in \mathbb{R}^m$, where m is the embedding dimension. Each user j is assigned with a vector of interest $\boldsymbol{u}_j \in \mathbb{R}^m$. To find realistic settings, we first observe the distribution of user embeddings, item embeddings, user biases, and item biases in pre-trained $R(\boldsymbol{r}|\boldsymbol{s})$ of MovieLens dataset, then use the same mean and variance to randomly sample each $\boldsymbol{v}_i$, $\boldsymbol{u}_j$, user bias b^u_j, item bias b^v_j, and global bias b. And the user's initial interest in d^i is given by:

$$\mathcal{I}_{\text{URM}}(d^i, j) = g(\boldsymbol{u}_j^T \boldsymbol{v}_i + b^u_j + b^v_i + b)$$

where g is a Sigmoid function. This basic model assumes independent point-wise interaction for each user-item pair and no other effect from the slate context.

Adding Positional Bias (URM_P): Items appeared at the previous positions of the slate are assumed to have a higher chance of receiving positive feedback than those in later positions. The reason for this design is that users may gradually lose their patience when further browsing the items [26]. In our setting, we first employ an average positional offset $\boldsymbol{b}^p = [0.2, 0.1, 0.0, -0.1, -0.2]$. And for each user, we draw personalized positional bias $\mathcal{B}(j) \sim \mathcal{N}(\boldsymbol{b}^p, \boldsymbol{\sigma}^2_u)$ where the variance $\boldsymbol{\sigma}^2_u = 0.2$. Then the final probability of click:

$$\mathcal{I}_{\text{URM\_P}}(d^i, j) = \text{clip}\big(\mathcal{I}_{\text{URM}}(d^i, j) + \lambda \mathcal{B}(j)_k, 0, 1\big)$$

where λ (set to 1.0 during experiment) controls how significant is the impact of positional bias on the user responses. The clip function ensures that the user's interest is within $[0, 1]$.

Adding Item Interactions (URM_P_MR): In [28], the authors assumed that item interactions are combinations of binary relations. Here we use a simple and easy-to-control multi-item relation model: First assume that a user's attention is altered when she sees the overall features of the slate:

$$\text{Atn}(\boldsymbol{s}, j) = g\left(\left(\frac{1}{K}\sum_{d^i \in \boldsymbol{s}} (\boldsymbol{v}_i)\right) \odot \boldsymbol{u}_j\right)$$

where $\odot$ denotes element-wise multiplication. Then the resulting attention is applied to each item to obtain the excursion:

$$\mathcal{M}(\boldsymbol{s}, j)_i = \text{Atn}(\boldsymbol{s}, j)^T \boldsymbol{v}_i$$

Add up everything so far gives the final probability of click:

$$\mathcal{I}_{\text{URM\_P\_MR}}(d^i, j) = \text{clip}\big(\mathcal{I}_{\text{URM}}(d^i, j) + \lambda \mathcal{B}(j)_k + \rho \mathcal{M}(\boldsymbol{s}, j)_i, 0, 1\big)$$

where coefficient ρ is introduced to control the significance of the item relation term. Though we found these simulation settings sufficient for our study, one may sue for more realistic and advanced simulation like [25] as complementary approaches.

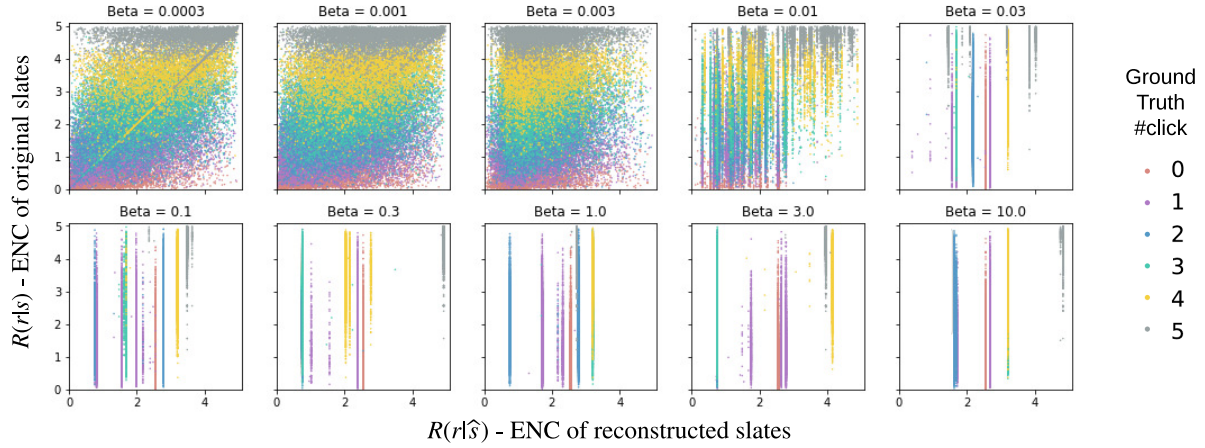

Figure 6: The reconstruction behavior of RCD on the entire Yoochoose dataset (including train, val, and test).

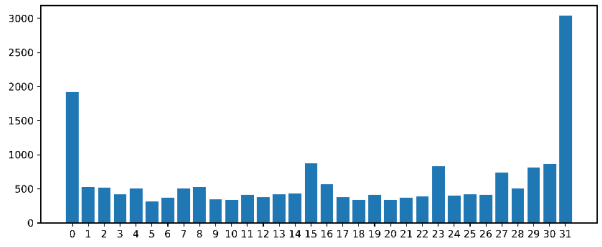

Figure 8: Slate response distribution of the preprocessed MovieLens 100K dataset with slate size $k = 5$.

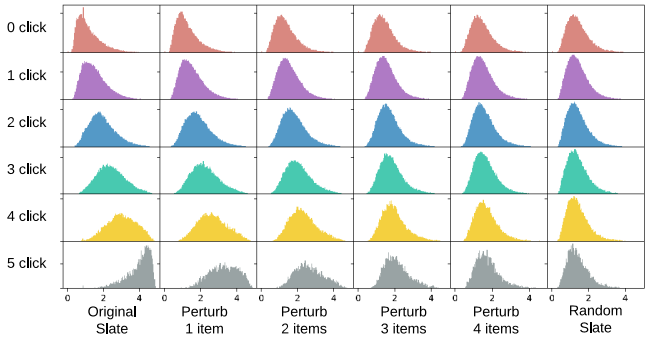

Figure 7: User response distribution under perturbation.

Table 3: Infeasible Evaluation on Test Set, see section D.1

	ML(User)	URM_P	URM_P_MR (ρ=0.5)	URM_P_MR (ρ=5.0)
D: Slate Hit Rate				
MF	**0.0330**	0.0069	0.0071	0.0092
NeuMF	0.0272	**0.0089**	**0.0088**	0.0068
MF-MMR	0.0092	0.0057	0.0078	0.0070
Non-Greedy MF	0.0283	0.0063	0.0072	0.0087
Non-Greedy NeuMF	0.0233	0.0077	0.0080	0.0072
List-CVAE	0.0041	0.0071	0.0079	**0.0093**
Non-Greedy List-CVAE	0.0056	0.0068	0.0078	0.0090
Pivot-CVAE (SGT-PI)	0.0043	0.0071	0.0069	0.0078
Pivot-CVAE (GT-SPI)	0.0131	0.0062	0.0072	0.0080
Pivot-CVAE (SGT-SPI)	0.0043	0.0070	0.0072	0.0074
E: Slate Recall				
MF	**0.0090**	0.0021	0.0024	0.0034
NeuMF	0.0079	**0.0029**	**0.0029**	0.0024
MF-MMR	0.0032	0.0018	0.0026	0.0025
Non-Greedy MF	0.0078	0.0019	0.0025	0.0032
Non-Greedy NeuMF	0.0072	0.0025	0.0026	0.0026
List-CVAE	0.0013	0.0024	**0.0029**	**0.0038**
Non-Greedy List-CVAE	0.0021	0.0022	0.0027	0.0035
Pivot-CVAE (SGT-PI)	0.0014	0.0023	0.0024	0.0026
Pivot-CVAE (GT-SPI)	0.0038	0.0020	0.0024	0.0028
Pivot-CVAE (SGT-SPI)	0.0013	0.0023	0.0025	0.0028

D.1 Stochastic vs. Deterministic

Compare to generative models, discriminative ranking models are deterministic and cannot explore the variety of slates, thus they are favored by ranking metrics on offline test set. We present the ranking performance on test set in Table 3-D and 3-E. For most datasets, different from the "ground truth" user response evaluated by environment $R(r|S)$, generative models (CVAE-based models) are stochastic and tend to explore more choices of good but unseen slates beyond the limited observation of the test set. On the other hand, ranking models like MF and NeuMF tend to focus on the best point that satisfies users the most, and perturbation of just one item does not severely harm the ranking metrics of the whole slate since the remaining items are also individually accurate. Interestingly, the gap between deterministic and stochastic models becomes less observable when we increase the proportion of item relations in the slates (URM_P $\rightarrow$ URM_MR), and generative models even start to outperform MF and NeuMF when $\rho = 5.0$. This shows that generative models are able to model whole-slate patterns for which MF and NeuMF often fail to learn.