

## A RELATED WORK

We highlight key related works to contextualize how PreferDiff fits within and contributes to the broader literature. Specifically, our work aligns with research on sequential recommendation and DMs based recommenders.

**Sequential Recommendation** have gained significant attention in both academia (Rendle, 2022; Liu et al., 2024) and industry (Wang et al., 2019; Fang et al., 2020) due to their ability to capture user preferences from historical interactions and recommend the next item. One common research line has focused on developing more efficient network architectures, such as GRU (Hidasi et al., 2016), convolutional neural networks (Tang & Wang, 2018), Transformer (Kang & McAuley, 2018; Fan et al., 2021), Bert4Rec (Devlin et al., 2019), and HSTU (Zhai et al., 2024). Another research line focuses on leveraging additional unsupervised signals (Xie et al., 2022; Wang et al., 2023a; Ren et al., 2024a) or reshaping sequential recommendation into other tasks such as retrieval (Rajput et al., 2023; Wang et al., 2024a) and language generation (Bao et al., 2023; Li et al., 2023b; Liao et al., 2024).

**DM-based Recommenders** have been explored in recent studies due to the powerful generative and generalization capabilities of DMs (DMs) (Lin et al., 2024). These recommenders either focus on modeling the distribution of the next item (e.g., (Yang et al., 2023b; Wang et al., 2024b; Li et al., 2024)), capture the probability distribution of user interactions (e.g., (Wang et al., 2023b; Zhao et al., 2024)), or focus on the distribution of time intervals between user behaviors (e.g., (Ma et al., 2024a)). However, existing approaches often rely on conventional objectives, such as mean squared error (MSE), or standard recommendation-specific objectives like Bayesian Personalized Ranking (BPR) (Rendle et al., 2009) and Cross Entropy (CE) (Klenitskiy & Vasilev, 2023). We argue that the former may diverge from the core objective of accurately modeling user preference distributions in recommendation tasks (Rendle, 2022), as DMs often lack an adequate understanding of negative items. While the latter leverages DMs’ noise resistance to mitigate noisy interactions in recommendations which might fall short of fully exploiting the generative and generalization capabilities of DMs.

## B SAMPLING ALGORITHM IN PREFERDIFF

We utilize DDIM (Song et al., 2021a) as the default sampler in PreferDiff, replacing the DDPM used in DreamRec, as we empirically find that DDIM is faster and performs better, requiring only a few denoising steps. Here, we briefly introduce how DDIM is employed in PreferDiff; Detailed derivations can be found in (Song et al., 2021a), and the code implementation is available at <https://anonymous.4open.science/r/PreferDiff>.

**Details.** Specifically, in PreferDiff, the training is to predict the original data  $\mathbf{e}_0$ . The sampling process should be reparameterized to predict  $\mathbf{e}_0$  directly instead of the noise  $\epsilon$ . Starting from the original DDIM update equation (Song et al., 2021a):

$$\mathbf{e}_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{\mathbf{e}_t - \sqrt{1 - \alpha_t} \epsilon_\theta(\mathbf{e}_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \epsilon_\theta(\mathbf{e}_t, t) + \sigma_t \mathbf{z}, \quad (13)$$

where  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $\sigma_t$  controls the stochasticity of the process, and  $\epsilon_\theta(\mathbf{e}_t, t)$  is the predicted noise at time step  $t$ .

In **PreferDiff**, since our model is trained to predict the original data  $\mathbf{e}_0$  directly, we use the relationship between  $\mathbf{e}_t$ ,  $\mathbf{e}_0$ , and the noise  $\epsilon$ :

$$\mathbf{e}_t = \sqrt{\alpha_t} \mathbf{e}_0 + \sqrt{1 - \alpha_t} \epsilon. \quad (14)$$

Solving for  $\epsilon$ , we obtain:

$$\epsilon = \frac{\mathbf{e}_t - \sqrt{\alpha_t} \mathbf{e}_0}{\sqrt{1 - \alpha_t}}. \quad (15)$$

Since  $\mathbf{e}_0$  is predicted by our model as  $\hat{\mathbf{e}}_0 = \mathcal{F}_\theta(\mathbf{e}_t, c, t)$ , we can estimate the noise as:

$$\hat{\epsilon}_\theta = \frac{\mathbf{e}_t - \sqrt{\alpha_t} \hat{\mathbf{e}}_0}{\sqrt{1 - \alpha_t}}. \quad (16)$$

Substituting  $\hat{\epsilon}_\theta$  back into the DDIM update equation and setting  $\sigma_t = 0$  for deterministic sampling, we get:

$$\mathbf{e}_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{\mathbf{e}_t - \sqrt{1 - \alpha_t} \hat{\epsilon}_\theta}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1}} \hat{\epsilon}_\theta \quad (17)$$

$$= \sqrt{\alpha_{t-1}} \hat{\mathbf{e}}_0 + \sqrt{1 - \alpha_{t-1}} \hat{\epsilon}_\theta. \quad (18)$$

This simplification allows us to update  $\mathbf{e}_{t-1}$  directly using the predicted  $\hat{\mathbf{e}}_0$  and  $\hat{\epsilon}_\theta$  without introducing additional randomness, thus making the sampling process deterministic and more efficient.

**Summary.** Therefore, the deterministic DDIM sampling steps in our inference algorithm are:

1. **Predict**  $\hat{\mathbf{e}}_0 = \mathcal{F}_\theta(\mathbf{e}_t, c, t)$ .
2. **Compute**  $\hat{\epsilon}_\theta = \frac{\mathbf{e}_t - \sqrt{\alpha_t} \hat{\mathbf{e}}_0}{\sqrt{1 - \alpha_t}}$ .
3. **Update**  $\mathbf{e}_{t-1} = \sqrt{\alpha_{t-1}} \hat{\mathbf{e}}_0 + \sqrt{1 - \alpha_{t-1}} \hat{\epsilon}_\theta$ .

By iteratively applying these steps, we can efficiently generate the predicted original data  $\hat{\mathbf{e}}_0$ . During inference, by setting  $\sigma_t = 0$ , we eliminate the noise term  $\sigma_t \mathbf{z}$  and focus solely on the deterministic components of the update rule. This results in faster convergence with fewer denoising steps while maintaining high-quality predictions. Detailed derivations and explanations of this reparameterization and the DDIM sampling process can be found in (Song et al., 2021a).

## C DETAILS ABOUT PREFERDIFF

### C.1 FROM RATINGS TO PROBABILITY DISTRIBUTION

$$\mathcal{L}_{\text{BPR}} = -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} [\log \sigma(f_\theta(\mathbf{e}_0^+ | \mathbf{c}) - f_\theta(\mathbf{e}_0^- | \mathbf{c}))], \quad (19)$$

The primary objective of equation 19 is to maximize the rating margin between positive items and sampled negative items. Here, we employ softmax normalization to transform the rating ranking into a log-likelihood ranking.

We begin by expressing the rating  $f_\theta(\mathbf{e}_0 | \mathbf{c})$  in terms of the probability distribution  $p_\theta(\mathbf{e}_0 | \mathbf{c})$ . This relationship is established through the following set of equations:

$$\begin{aligned} p_\theta(\mathbf{e}_0 | \mathbf{c}) &= \frac{\exp(f_\theta(\mathbf{e}_0 | \mathbf{c}))}{Z_\theta}, \\ \log p_\theta(\mathbf{e}_0 | \mathbf{c}) &= f_\theta(\mathbf{e}_0 | \mathbf{c}) - \log Z_\theta, \\ f_\theta(\mathbf{e}_0 | \mathbf{c}) &= \log p_\theta(\mathbf{e}_0 | \mathbf{c}) + \log Z_\theta. \end{aligned} \quad (20)$$

Substituting equation 20 into equation 19 yields the BPR loss expressed solely in terms of the probability distributions of positive and negative items.

$$\begin{aligned}
\mathcal{L}_{\text{BPR-Diff}} &= -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \underbrace{f_\theta(\mathbf{e}_0^+ | \mathbf{c})}_{\text{rating of Positive Item}} - \underbrace{f_\theta(\mathbf{e}_0^- | \mathbf{c})}_{\text{rating of Negative Item}} \right) \right] \\
&= -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \underbrace{\log p_\theta(\mathbf{e}_0^+ | \mathbf{c}) + \log Z_\theta}_{\text{From equation 20}} - \underbrace{\log p_\theta(\mathbf{e}_0^- | \mathbf{c}) - \log Z_\theta}_{\text{From equation 20}} \right) \right] \\
&= -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \log p_\theta(\mathbf{e}_0^+ | \mathbf{c}) - \log p_\theta(\mathbf{e}_0^- | \mathbf{c}) + \underbrace{\log Z_\theta - \log Z_\theta}_{=0} \right) \right] \\
&= -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \log \frac{p_\theta(\mathbf{e}_0^+ | \mathbf{c})}{p_\theta(\mathbf{e}_0^- | \mathbf{c})} \right) \right].
\end{aligned} \tag{21}$$

## C.2 CONNECTING THE RATING FUNCTION TO THE SCORE FUNCTION

In this subsection, we establish the relationship between the rating function  $f_\theta(\mathbf{e}_0 | \mathbf{c})$  and the score function in the context of score-based DMs. Specifically, we demonstrate that the gradient of the rating function with respect to the item embedding  $\mathbf{e}_0$  is equivalent to the score function  $\nabla_{\mathbf{e}_0} \log p_\theta(\mathbf{e}_0 | \mathbf{c})$ .

Starting from Equation equation 20:

$$f_\theta(\mathbf{e}_0 | \mathbf{c}) = \log p_\theta(\mathbf{e}_0 | \mathbf{c}) + \log Z_\theta, \tag{22}$$

where  $Z_\theta$  is the partition function:

$$Z_\theta = \int \exp(f_\theta(\mathbf{e} | \mathbf{c})) d\mathbf{e}. \tag{23}$$

### DERIVATIVE OF THE RATING FUNCTION WITH RESPECT TO $\mathbf{e}_0$

Taking the gradient of Equation equation 22 with respect to  $\mathbf{e}_0$ , we have:

$$\nabla_{\mathbf{e}_0} f_\theta(\mathbf{e}_0 | \mathbf{c}) = \nabla_{\mathbf{e}_0} \log p_\theta(\mathbf{e}_0 | \mathbf{c}) + \nabla_{\mathbf{e}_0} \log Z_\theta. \tag{24}$$

Since the partition function  $Z_\theta$  is obtained by integrating over all possible item embeddings  $\mathbf{e}$ , and does not depend on the specific  $\mathbf{e}_0$ , its gradient with respect to  $\mathbf{e}_0$  is zero:

$$\nabla_{\mathbf{e}_0} \log Z_\theta = 0. \tag{25}$$

Therefore, Equation equation 24 simplifies to:

$$\nabla_{\mathbf{e}_0} f_\theta(\mathbf{e}_0 | \mathbf{c}) = \nabla_{\mathbf{e}_0} \log p_\theta(\mathbf{e}_0 | \mathbf{c}). \tag{26}$$

**Definition of the Score Function** In score-based DMs, the **score function** is defined as the gradient of the log-probability density with respect to the data point  $\mathbf{e}_0$ :

$$\mathbf{s}_\theta(\mathbf{e}_0, \mathbf{c}) \triangleq \nabla_{\mathbf{e}_0} \log p_\theta(\mathbf{e}_0 | \mathbf{c}). \tag{27}$$

Comparing Equations equation 26 and equation 27, we find that:

$$\nabla_{\mathbf{e}_0} f_\theta(\mathbf{e}_0 | \mathbf{c}) = \mathbf{s}_\theta(\mathbf{e}_0, \mathbf{c}). \tag{28}$$

This reveals that the gradient of the rating function with respect to the item embedding  $\mathbf{e}_0$  is exactly the score function of the probability distribution  $p_\theta(\mathbf{e}_0 | \mathbf{c})$ . Score-based DMs Song et al. (2021b) utilize the score function  $\mathbf{s}_\theta(\mathbf{e}_0, \mathbf{c})$  to define the reverse diffusion process. In these models, the data generation process involves integrating the score function over time to recover the data distribution from noise. Intuitively, we can utilize  $\nabla_{\mathbf{e}_0} f_\theta(\mathbf{e}_0 | \mathbf{c})$  to sample item embeddings with high ratings

through Langevin dynamics (Song & Ermon, 2020) given certain user historical conditions. Therefore, it bridges the objective of recommendation with generative modeling in DMs.

**Connection to Our Loss Function.** Our BPR-Diff loss function, as expressed in Equation equation 21, involves the log-ratio of the probabilities of positive and negative items:

$$\mathcal{L}_{\text{BPR-Diff}} = -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \log \frac{p_\theta(\mathbf{e}_0^+ | \mathbf{c})}{p_\theta(\mathbf{e}_0^- | \mathbf{c})} \right) \right]. \quad (29)$$

Using the equivalence between the rating function and the log-probability (from Equation equation 22), the loss function can also be seen as a function of the rating differences:

$$\mathcal{L}_{\text{BPR-Diff}} = -\mathbb{E} [\log \sigma (f_\theta(\mathbf{e}_0^+ | \mathbf{c}) - f_\theta(\mathbf{e}_0^- | \mathbf{c}))]. \quad (30)$$

**Gradient of the Loss with Respect to  $\mathbf{e}_0^+$ .** Taking the gradient of the loss function with respect to the positive item embedding  $\mathbf{e}_0^+$ , we get:

$$\nabla_{\mathbf{e}_0^+} \mathcal{L}_{\text{BPR-Diff}} = -\mathbb{E} [\sigma(-s) \cdot \nabla_{\mathbf{e}_0^+} f_\theta(\mathbf{e}_0^+ | \mathbf{c})], \quad (31)$$

where  $s = f_\theta(\mathbf{e}_0^+ | \mathbf{c}) - f_\theta(\mathbf{e}_0^- | \mathbf{c})$ .

Similarly, for the negative item embedding  $\mathbf{e}_0^-$ :

$$\nabla_{\mathbf{e}_0^-} \mathcal{L}_{\text{BPR-Diff}} = \mathbb{E} [\sigma(-s) \cdot \nabla_{\mathbf{e}_0^-} f_\theta(\mathbf{e}_0^- | \mathbf{c})]. \quad (32)$$

These gradients indicate that the loss function encourages:

- Increasing the rating  $f_\theta(\mathbf{e}_0^+ | \mathbf{c})$  of the positive item by moving  $\mathbf{e}_0^+$  in the direction of  $\nabla_{\mathbf{e}_0^+} f_\theta$ .
- Decreasing the rating  $f_\theta(\mathbf{e}_0^- | \mathbf{c})$  of the negative item by moving  $\mathbf{e}_0^-$  opposite to  $\nabla_{\mathbf{e}_0^-} f_\theta$ .

### C.3 DERIVATION THE VARIATIONAL UPPER BOUND

In this section, we provide a comprehensive derivation of the upper bound for the proposed  $\mathcal{L}_{\text{BPR-Diff}}$ . We focus particularly on the steps involving the Kullback-Leibler divergence, leading to the final loss function used for training.

#### Assumptions and Definitions:

- $\mathbf{e}_0^+$  and  $\mathbf{e}_0^-$  represent the embeddings of the positive and negative items, respectively.
- $\mathbf{e}_t^+$  and  $\mathbf{e}_t^-$  are the noisy embeddings at timestep  $t$  for the positive and negative items, obtained via the forward diffusion process.
- $\mathbf{c}$  denotes the historical item sequence for a user.
- $q(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{e}_0)$  is the posterior distribution in the forward diffusion process.
- $p_\theta(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{c})$  is the reverse diffusion process modeled by our neural network  $\mathcal{F}_\theta$ .
- $\mathcal{M}(\mathbf{c})$  is a mapping function that encodes the historical context  $\mathbf{c}$  into a suitable representation for conditioning.
- $\sigma(\cdot)$  is the sigmoid function.
- $\beta_t$ ,  $\alpha_t$ , and  $\bar{\alpha}_t$  are predefined constants in the diffusion schedule.

Starting from equation 4 in the main text, we have:

$$\mathcal{L}_{\text{BPR-Diff}}(\theta) = -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( \log \mathbb{E}_{q(\mathbf{e}_{1:T}^+ | \mathbf{e}_0^+)} \left[ \frac{p_\theta(\mathbf{e}_{0:T}^+ | \mathbf{c})}{q(\mathbf{e}_{1:T}^+ | \mathbf{e}_0^+)} \right] - \log \mathbb{E}_{q(\mathbf{e}_{1:T}^- | \mathbf{e}_0^-)} \left[ \frac{p_\theta(\mathbf{e}_{0:T}^- | \mathbf{c})}{q(\mathbf{e}_{1:T}^- | \mathbf{e}_0^-)} \right] \right) \right]. \quad (33)$$

To address the intractability of directly computing the expectations inside the logarithms, we apply Jensen’s inequality, which states that for a convex function  $f$ , we have  $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ . Recognizing that  $-\log \sigma(x)$  is convex in  $x$ , we obtain an upper bound:

$$\mathcal{L}_{\text{BPR-Diff}}(\theta) \leq -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \mathbb{E}_{q(\mathbf{e}_{1:T}^+ | \mathbf{e}_0^+), q(\mathbf{e}_{1:T}^- | \mathbf{e}_0^-)} \left[ \log \sigma \left( \underbrace{\log \left[ \frac{p_\theta(\mathbf{e}_{0:T}^+ | \mathbf{c})}{q(\mathbf{e}_{1:T}^+ | \mathbf{e}_0^+)} \right]}_{(a)} - \underbrace{\log \left[ \frac{p_\theta(\mathbf{e}_{0:T}^- | \mathbf{c})}{q(\mathbf{e}_{1:T}^- | \mathbf{e}_0^-)} \right]}_{(b)} \right) \right]. \quad (34)$$

The terms (a) and (b) represent the variational lower bounds of the log-likelihoods for the positive and negative items, respectively. According to the properties of DMs (Ho et al., 2020), these terms can be related to the evidence lower bound (ELBO). Specifically, for any item  $\mathbf{e}_0$ , we have:

$$\log p_\theta(\mathbf{e}_0 | \mathbf{c}) \geq \mathbb{E}_{q(\mathbf{e}_{1:T} | \mathbf{e}_0)} \left[ \log \left( \frac{p_\theta(\mathbf{e}_{0:T} | \mathbf{c})}{q(\mathbf{e}_{1:T} | \mathbf{e}_0)} \right) \right] = -\mathcal{L}_{\text{ELBO}}(\theta; \mathbf{e}_0, \mathbf{c}). \quad (35)$$

Substituting equation 35 into equation 34, we get:

$$\mathcal{L}_{\text{BPR-Diff}}(\theta) \leq -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} [\log \sigma (-\mathcal{L}_{\text{ELBO}}(\theta; \mathbf{e}_0^+, \mathbf{c}) + \mathcal{L}_{\text{ELBO}}(\theta; \mathbf{e}_0^-, \mathbf{c}))]. \quad (36)$$

The ELBO for each item can be decomposed into a sum over timesteps  $t$ :

$$\mathcal{L}_{\text{ELBO}}(\theta; \mathbf{e}_0, \mathbf{c}) = \sum_{t=1}^T \mathbb{E}_{q(\mathbf{e}_t | \mathbf{e}_0)} [D_{\text{KL}}(q(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{e}_0) \| p_\theta(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{c}))] + C, \quad (37)$$

where  $C$  is a constant independent of  $\theta$ .

Substituting equation 37 back into equation 36, we obtain:

$$\mathcal{L}_{\text{BPR-Diff}}(\theta) \leq -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c})} \left[ \log \sigma \left( - \left( \sum_{t=1}^T \mathbb{E}_{q(\mathbf{e}_t^+ | \mathbf{e}_0^+)} [D_{\text{KL}}(q(\mathbf{e}_{t-1}^+ | \mathbf{e}_t^+, \mathbf{e}_0^+) \| p_\theta(\mathbf{e}_{t-1}^+ | \mathbf{e}_t^+, \mathbf{c}))] \right. \right. \right. \\ \left. \left. \left. - \sum_{t=1}^T \mathbb{E}_{q(\mathbf{e}_t^- | \mathbf{e}_0^-)} [D_{\text{KL}}(q(\mathbf{e}_{t-1}^- | \mathbf{e}_t^-, \mathbf{e}_0^-) \| p_\theta(\mathbf{e}_{t-1}^- | \mathbf{e}_t^-, \mathbf{c}))] + C_1 \right) \right) \right], \quad (38)$$

where  $C_1$  aggregates constants and is independent of  $\theta$ .

Now, we focus on the KL divergence terms. In DMs, both  $q(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{e}_0)$  and  $p_\theta(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{c})$  are Gaussian distributions (Ho et al., 2020). Specifically, for the forward process  $q$  and the reverse process  $p_\theta$ , we have:

$$q(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{e}_0) = \mathcal{N}(\mathbf{e}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{e}_t, \mathbf{e}_0), \tilde{\beta}_t \mathbf{I}), \quad (39)$$

$$p_\theta(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{c}) = \mathcal{N}(\mathbf{e}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{e}_t, t, \mathbf{c}), \beta_t \mathbf{I}), \quad (40)$$

where  $\tilde{\boldsymbol{\mu}}_t(\mathbf{e}_t, \mathbf{e}_0)$  is the mean of the posterior  $q(\mathbf{e}_{t-1} | \mathbf{e}_t, \mathbf{e}_0)$ ,  $\tilde{\beta}_t$  is the variance, and  $\beta_t$  is the variance schedule for the reverse process.

The KL divergence between two Gaussian distributions can be computed as:

$$D_{\text{KL}}(q \| p_\theta) = \frac{1}{2} \left( \text{tr}(\beta_t^{-1} \tilde{\beta}_t \mathbf{I}) + (\boldsymbol{\mu}_\theta - \tilde{\boldsymbol{\mu}}_t)^\top \beta_t^{-1} \mathbf{I} (\boldsymbol{\mu}_\theta - \tilde{\boldsymbol{\mu}}_t) - k + \ln \left( \frac{\det(\beta_t \mathbf{I})}{\det(\tilde{\beta}_t \mathbf{I})} \right) \right), \quad (41)$$

where  $k$  is the dimensionality of the Gaussian distributions (i.e., the embedding dimension).

Assuming that  $\tilde{\beta}_t = \beta_t$  (Ho et al., 2020), the trace term simplifies to  $k$ , and the determinant term becomes  $\ln(1) = 0$ . Therefore, the KL divergence simplifies to:

$$D_{\text{KL}}(q \| p_\theta) = \frac{1}{2\beta_t} \|\boldsymbol{\mu}_\theta - \tilde{\boldsymbol{\mu}}_t\|_2^2. \quad (42)$$

Next, we define the network prediction  $\mu_\theta$  and relate it to the mean  $\tilde{\mu}_t$  from the forward process.

#### Relationship between $\tilde{\mu}_t$ and $\mathbf{e}_0$ :

The mean  $\tilde{\mu}_t$  is given by:

$$\tilde{\mu}_t(\mathbf{e}_t, \mathbf{e}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\mathbf{e}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{e}_t, \quad (43)$$

where  $\alpha_t = 1 - \beta_t$ , and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ . In practice, it is common to predict  $\mathbf{e}_0$  directly using the neural network  $\mathcal{F}_\theta$ :

$$\hat{\mathbf{e}}_0 = \mathcal{F}_\theta(\mathbf{e}_t, t, \mathcal{M}(\mathbf{c})). \quad (44)$$

Given  $\hat{\mathbf{e}}_0$ , we can compute  $\mu_\theta$  as:

$$\mu_\theta(\mathbf{e}_t, t, \mathbf{c}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\hat{\mathbf{e}}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{e}_t. \quad (45)$$

Substituting equations equation 43 and equation 45 into equation 42, we have:

$$D_{\text{KL}}(q \| p_\theta) = \frac{1}{2\beta_t} \|\mu_\theta - \tilde{\mu}_t\|_2^2 = \frac{1}{2\beta_t} \left\| \left( \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}(\hat{\mathbf{e}}_0 - \mathbf{e}_0) \right) \right\|_2^2 = \frac{(\sqrt{\bar{\alpha}_{t-1}}\beta_t)^2}{2\beta_t^2(1 - \bar{\alpha}_t)^2} \|\hat{\mathbf{e}}_0 - \mathbf{e}_0\|_2^2. \quad (46)$$

Simplifying the constants, we observe that the coefficient reduces to a constant factor dependent on  $t$ , which we can denote as  $\lambda_t$ :

$$\lambda_t = \frac{(\sqrt{\bar{\alpha}_{t-1}}\beta_t)^2}{2\beta_t^2(1 - \bar{\alpha}_t)^2} = \frac{\bar{\alpha}_{t-1}}{2(1 - \bar{\alpha}_t)^2}. \quad (47)$$

Therefore, the KL divergence becomes:

$$D_{\text{KL}}(q \| p_\theta) = \lambda_t \|\hat{\mathbf{e}}_0 - \mathbf{e}_0\|_2^2. \quad (48)$$

Since  $\lambda_t$  is independent of  $\theta$  and depends only on  $t$ , when we sum over all timesteps and average over  $t$ , this term becomes proportional to the mean squared error between  $\hat{\mathbf{e}}_0$  and  $\mathbf{e}_0$ .

#### Equivalence of MSE and Cosine Error for Unit Norm Vectors:

Alternatively, to mitigate sensitivity to vector norms and dimensionality (Friedman, 1997; Hou et al., 2022b) (the recommendation performance of PreferDiff is competitive when embedding size is higher), we can use the cosine error as the distance measure. The cosine similarity between  $\hat{\mathbf{e}}_0$  and  $\mathbf{e}_0$  is given by:

$$\cos(\hat{\mathbf{e}}_0, \mathbf{e}_0) = \frac{\hat{\mathbf{e}}_0^\top \mathbf{e}_0}{\|\hat{\mathbf{e}}_0\|_2 \|\mathbf{e}_0\|_2}. \quad (49)$$

The cosine error is then:

$$S(\hat{\mathbf{e}}_0, \mathbf{e}_0) = 1 - \cos(\hat{\mathbf{e}}_0, \mathbf{e}_0). \quad (50)$$

Actually, when both  $\hat{\mathbf{e}}_0$  and  $\mathbf{e}_0$  are normalized to have unit norm (i.e.,  $\|\hat{\mathbf{e}}_0\|_2 = \|\mathbf{e}_0\|_2 = 1$ ), the mean squared error and the cosine error are directly related. Specifically, the squared Euclidean distance between two unit vectors is:

$$\|\hat{\mathbf{e}}_0 - \mathbf{e}_0\|_2^2 = (\hat{\mathbf{e}}_0 - \mathbf{e}_0)^\top (\hat{\mathbf{e}}_0 - \mathbf{e}_0) = \|\hat{\mathbf{e}}_0\|_2^2 + \|\mathbf{e}_0\|_2^2 - 2\hat{\mathbf{e}}_0^\top \mathbf{e}_0 = 2(1 - \cos(\hat{\mathbf{e}}_0, \mathbf{e}_0)). \quad (51)$$

Thus, under the unit norm constraint, minimizing the MSE is equivalent to minimizing the cosine error up to a constant factor of 2. This shows that both distance measures capture the same notion of similarity in this case. Substituting the KL divergence approximation back into equation 38, and considering both positive and negative items, we simplify the expression:

$$\mathcal{L}_{\text{BPR-Diff}}(\theta) \leq -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c}), t \sim U(1, T)} \left[ \log \sigma \left( - \left( \underbrace{S(\hat{\mathbf{e}}_0^+, \mathbf{e}_0^+)}_{\text{Positive item error}} - \underbrace{S(\hat{\mathbf{e}}_0^-, \mathbf{e}_0^-)}_{\text{Negative item error}} \right) \right) \right], \quad (52)$$

where  $\hat{\mathbf{e}}_0^+ = \mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c}))$  and  $\hat{\mathbf{e}}_0^- = \mathcal{F}_\theta(\mathbf{e}_t^-, t, \mathcal{M}(\mathbf{c}))$ .

Equation 52 represents our final trainable objective:

$$\mathcal{L}_{\text{Upper}}(\theta) = -\mathbb{E}_{(\mathbf{e}_0^+, \mathbf{e}_0^-, \mathbf{c}), t \sim U(1, T)} [\log \sigma(- (S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - S(\mathcal{F}_\theta(\mathbf{e}_t^-, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^-)))] . \quad (53)$$

**Explanation.** This objective encourages the model to minimize the distance between the predicted embedding and the true embedding for the positive item while maximizing the distance for the negative item, effectively widening the gap between them in the latent space. By doing so, we enhance the personalized ranking capability of the model.

**Summary.** By minimizing  $\mathcal{L}_{\text{Upper}}(\theta)$ , we implicitly minimize the original  $\mathcal{L}_{\text{BPR-Diff}}(\theta)$  due to the application of Jensen’s inequality. This aligns the training objective with the goal of improving personalized ranking by leveraging DMs within the BPR.

#### C.4 EXTEND INTO MULTIPLE NEGATIVE SAMPLES

In this section, we provide a detailed derivation of the inequality  $\mathcal{L}_{\text{BPR-Diff-V}} \leq \mathcal{L}_{\text{BPR-Diff-C}}$ , under the assumption that  $\mathcal{F}_\theta$  and  $S$  are convex functions.

##### Definitions and Assumptions

We define:

- $\mathcal{F}_\theta(\mathbf{e}_t, t, \mathcal{M}(\mathbf{c}))$ : the denoising function at time step  $t$ , parameterized by  $\theta$ , conditioned on context  $\mathcal{M}(\mathbf{c})$ .
- $S(\mathbf{a}, \mathbf{b})$ : a measure function quantifying the discrepancy between vectors  $\mathbf{a}$  and  $\mathbf{b}$ , such as Mean Squared Error (MSE).
- $\sigma(\cdot)$ : the sigmoid function.

Assume that:

- $\mathcal{F}_\theta$  is convex with respect to its input  $\mathbf{e}_t$ .
- $S$  is convex with respect to both of its inputs.

Starting with the definition of  $\mathcal{L}_{\text{BPR-Diff-V}}$ :

$$\mathcal{L}_{\text{BPR-Diff-V}} = -\log \sigma \left( -V \left( S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - \frac{1}{V} \sum_{v=1}^V S(\mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^{-v}) \right) \right) . \quad (54)$$

Similarly, for  $\mathcal{L}_{\text{BPR-Diff-C}}$ :

$$\mathcal{L}_{\text{BPR-Diff-C}} = -\log \sigma \left( -V \left( S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - S(\mathcal{F}_\theta(\tilde{\mathbf{e}}_t^-, t, \mathcal{M}(\mathbf{c})), \tilde{\mathbf{e}}_0^-) \right) \right) , \quad (55)$$

where we have defined the centroids:

$$\tilde{\mathbf{e}}_t^- = \frac{1}{V} \sum_{v=1}^V \mathbf{e}_t^{-v}, \quad \tilde{\mathbf{e}}_0^- = \frac{1}{V} \sum_{v=1}^V \mathbf{e}_0^{-v} . \quad (56)$$

Our aim is to show that  $\mathcal{L}_{\text{BPR-Diff-V}} \leq \mathcal{L}_{\text{BPR-Diff-C}}$ .

First, consider the term:

$$D_V = S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - \frac{1}{V} \sum_{v=1}^V S(\mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^{-v}) . \quad (57)$$

By the convexity of  $S$ , we have:

$$\frac{1}{V} \sum_{v=1}^V S(\mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^{-v}) \leq S\left(\underbrace{\frac{1}{V} \sum_{v=1}^V \mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c}))}_{\text{Convex combination of } \mathcal{F}_\theta(\mathbf{e}_t^{-v})}, \underbrace{\frac{1}{V} \sum_{v=1}^V \mathbf{e}_0^{-v}}_{\tilde{\mathbf{e}}_0^-}\right). \quad (58)$$

Next, using the convexity of  $\mathcal{F}_\theta$ , we have:

$$\mathcal{F}_\theta(\tilde{\mathbf{e}}_t^-, t, \mathcal{M}(\mathbf{c})) \leq \underbrace{\frac{1}{V} \sum_{v=1}^V \mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c}))}_{\text{Convex combination}}. \quad (59)$$

Combining equation 58 and equation 59, and recognizing that  $S$  is non-decreasing with respect to its first argument, we get:

$$\frac{1}{V} \sum_{v=1}^V S(\mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^{-v}) \leq S(\mathcal{F}_\theta(\tilde{\mathbf{e}}_t^-, t, \mathcal{M}(\mathbf{c})), \tilde{\mathbf{e}}_0^-). \quad (60)$$

Therefore, we have:

$$D_V = S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - \frac{1}{V} \sum_{v=1}^V S(\mathcal{F}_\theta(\mathbf{e}_t^{-v}, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^{-v}) \quad (61)$$

$$\geq S(\mathcal{F}_\theta(\mathbf{e}_t^+, t, \mathcal{M}(\mathbf{c})), \mathbf{e}_0^+) - S(\mathcal{F}_\theta(\tilde{\mathbf{e}}_t^-, t, \mathcal{M}(\mathbf{c})), \tilde{\mathbf{e}}_0^-) = D_C. \quad (62)$$

Since  $D_V \geq D_C$ , it follows that:

$$-VD_V \leq -VD_C. \quad (63)$$

Applying the monotonicity of the  $\log \sigma(\cdot)$  function (since  $\sigma$  is an increasing function and  $\log$  is monotonic), we have:

$$\mathcal{L}_{\text{BPR-Diff-V}} = -\log \sigma(-VD_V) \leq -\log \sigma(-VD_C) = \mathcal{L}_{\text{BPR-Diff-C}}. \quad (64)$$

Therefore, we have shown that:

$$\mathcal{L}_{\text{BPR-Diff-V}} \leq \mathcal{L}_{\text{BPR-Diff-C}}. \quad (65)$$

**Explanation.** This inequality implies that minimizing  $\mathcal{L}_{\text{BPR-Diff-C}}$  effectively minimizes an upper bound of  $\mathcal{L}_{\text{BPR-Diff-V}}$ , leading to an efficient increase in the likelihood of positive items while distancing them from the centroid of negative items. Notably, although the assumption of convexity is difficult to satisfy in practice, the aforementioned method still empirically achieves strong results than one negative item.

## D EXPERIMENTS

### D.1 DATASETS PREPROCESSING IN USER SPLITTING SETTING

Following prior works (Yang et al., 2023a;b), we adopt the user-splitting setting, which has been shown to effectively prevent information leakage in test sets (Ji et al., 2023). Specifically, we first

**Algorithm 2** Inference Phase of PreferDiff

---

```

1: Input: Trained parameters  $\theta$ , Sequence encoder  $\mathcal{M}(\cdot)$ , test dataset  $\mathcal{D}_{\text{test}} = \{(\mathbf{e}_0, \mathbf{c})\}_{n=1}^{|\mathcal{D}_{\text{test}}|}$ , total
   steps  $T$ , DDIM steps  $S$ , guidance weight  $w$ , variance schedules  $\{\alpha_t\}_{t=1}^T$ 
2: Output: Predicted next item  $\hat{\mathbf{e}}_0$ 
3:  $\mathbf{c} \sim \mathcal{D}_{\text{test}}$  ▷ Sample user historical sequence from testing dataset.
4:  $\mathbf{e}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  ▷ Sample standard Gaussian noise.
5: for  $s = S, \dots, 1$  do ▷ Denoise over  $S$  DDIM steps.
6:    $t = \lfloor s \times (T/S) \rfloor$  ▷ Map DDIM step  $s$  to original step  $t$ .
7:   With probability  $p_u$ :  $\mathcal{M}(\mathbf{c}) = \Phi$  ▷ Set unconditional condition with probability  $p_u$ .
8:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $s > 1$  else  $\mathbf{z} = \mathbf{0}$  ▷ Sample noise if not final step.
9:    $\hat{\mathbf{e}}_0 = (1 + w)\mathcal{F}_\theta(\hat{\mathbf{e}}_t, \mathcal{M}(\mathbf{c}), t) - w\mathcal{F}_\theta(\hat{\mathbf{e}}_t, \Phi, t)$  ▷ Apply classifier-free guidance.
10:   $\hat{\epsilon}_\theta = \frac{\hat{\mathbf{e}}_t - \sqrt{\alpha_t}\hat{\mathbf{e}}_0}{\sqrt{1 - \alpha_t}}$  ▷ Compute predicted noise.
11:   $\hat{\mathbf{e}}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\mathbf{e}}_0 + \sqrt{1 - \alpha_{t-1}}\hat{\epsilon}_\theta$  ▷ DDIM update step when  $\sigma_t = 0$ .
12: end for
13: return  $\hat{\mathbf{e}}_0$ 

```

---

Table 5: Detailed Statistics of Datasets after Preprocessing.

Datasets	Fully Trained Recommendation			General Sequential Recommendation				
	Sports	Beauty	Toys	Pretraining	Validation	CDs	Movies	Steam
#Sequences	35,598	22,363	19,412	746,688	101,501	112,379	297,529	39,795
#Items	18,357	12,101	11,924	68,668	8,623	15,520	25,925	9,265
#Interactions	256,598	162,150	138,444	3,258,523	452,415	457,589	2,053,497	437,733

sort all sequences chronologically for each dataset, then split the data into training, validation, and test sets with an 8:1:1 ratio, while preserving the last 10 interactions as the historical sequence.

**Amazon 2014**<sup>1</sup>. Here, we choose three public real-world benchmarks (i.e., Sports, Beauty and Toys) which has been widely utilized in recent studies (Rajput et al., 2023). Here, we utilize the common five-core datasets (Hou et al., 2022a), filtering out users and items with fewer than five interactions across all datasets. Following previous work (Yang et al., 2023b), we set the maximized length user interaction sequence as 10.

**Amazon 2018**<sup>2</sup>. Following prior works (Hou et al., 2022a; Li et al., 2023a), we select five distinct product review categories—namely, “Automotive,” “Electronics,” “Grocery and Gourmet Food,” “Musical Instruments,” and “Tools and Home Improvement”—as pretraining datasets. “Cell Phones and Accessories” is used as the validation set for early stopping. In line with previous research (Yang et al., 2023b), we filter out items with fewer than 20 interactions and user interaction sequences shorter than 5, capping the maximum length of each user’s interaction sequence at 10.

**Steam** is a game review dataset collected from Steam<sup>3</sup>. Due to the large number of game reviews, we filter out users and items with fewer than 20 interactions.

## D.2 IMPLEMENTATION DETAILS

For a fair comparison, all experiments are conducted in PyTorch using a single Tesla V100-SXM3-32GB GPU and an Intel(R) Xeon(R) Gold 6248R CPU. We optimize all methods using the AdamW optimizer and all models’ parameters are initialized with Xavier initialization. We fix the embedding dimension to 64 for all models except DM-based recommenders, as the latter only demonstrate strong performance with higher embedding dimensions, as discussed in Section 4.3. Since our focus is not on network architecture and for fair comparison, we adopt a lightweight configuration for baseline models that employ a Transformer backbone<sup>4</sup>, using a single layer with two attention heads. **Notably, all baselines, unless otherwise specified, use cross-entropy as the loss function**, as recent studies (Klenitskiy & Vasilev, 2023; Zhai et al., 2023) have demonstrated its effectiveness.

<sup>1</sup><https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html>

<sup>2</sup>[https://cseweb.ucsd.edu/~jmcauley/datasets/amazon\\_v2/](https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/)

<sup>3</sup><https://github.com/kang205/SASRec>

<sup>4</sup><https://github.com/YangZhengyi98/DreamRec/>

For PerferDiff, for each user sequence, we treat the other next-items (a.k.a., labels) in the same batch as negative samples. We set the default diffusion timestep to 2000, DDIM step as 20,  $p_u = 0.1$ , and the  $\beta$  linearly increase in the range of  $[1e^{-4}, 0.02]$  for all DM-based sequential recommenders (e.g., DreamRec). We empirically find that tuning these parameters may lead to better recommendation performance. However, as this is not the focus of the paper, we do not elaborate on it.

The other hyperparameter (e.g., learning rate) search space for PreferDiff and the baseline models is provided in Table 11, while the best hyperparameters for PreferDiff are listed in Table 12.

### D.3 BASELINES OF SEQUENTIAL RECOMMENDATION

Traditional sequential recommenders:

- **GRU4Rec** (Hidasi et al., 2016) adopts RNNs to model user behavior sequences for session-based recommendations. Here, following the previous work (Kang & McAuley, 2018; Yang et al., 2023b), we treat each user’s interaction sequence as a session.

- **SASRec** (Kang & McAuley, 2018) adopts a directional self-attention network to model the user user behavior sequences.

- **Bert4Rec** (Sun et al., 2019) adapts the original text-based BERT model with the cloze objective for modeling user behavior sequences. We adopt the implementation of mask from (Ren et al., 2024b)

Contrastive learning based sequential recommenders:

- **CL4SRec** (Xie et al., 2022) incorporates the contrastive learning with the transformer-based sequential recommendation model to obtain more robust results. We adopt the implementation <sup>5</sup> from (Ren et al., 2024b).

Generative sequential recommenders:

- **TIGER** (Rajput et al., 2023) introduces codebook-based identifiers through RQ-VAE, which quantizes semantic information into code sequences for generative recommendation. Since the source code is unavailable, we implement it using the HuggingFace and Transformers APIs, following the original paper by utilizing T5 (Ni et al., 2022) as the backbone. For quantization, we employ FAISS (Johnson et al., 2019), which is widely used <sup>6</sup> in recent studies of recommendation (Hou et al., 2023).

DM-based sequential recommenders:

- **DiffRec** (Wang et al., 2023b) introduces the application of diffusion on user interaction vectors (i.e., multi-hot vectors) for collaborative recommendation, where “1” denotes a positive interaction and “0” indicates a potential negative interaction. We adopt the author’s public implementation <sup>7</sup>.

- **DreamRec** (Yang et al., 2023b) uses the historical interaction sequence as conditional guiding information for the diffusion model to enable personalized recommendations and utilize MSE as the training objective. We adopt the author’s public implementation <sup>8</sup>.

- **DiffuRec** (Li et al., 2024) introduces the DM to reconstruct target item embedding from a Transformer backbone with the user’s historical interaction behaviors and utilize CE as the training objective. We adopt the author’s public implementation <sup>9</sup>.

Text-based sequential recommenders:

- **MoRec** (Yuan et al., 2023) utilizes item features from text descriptions or images, encoded using a text encoder or vision encoder, and applies dimensional transformation to match the appropriate dimension for recommendation. Here, we utilize the OpenAI-3-large embeddings, SASRec as backbone and transform the dimension to 64.

<sup>5</sup><https://github.com/HKUDS/SSLRec/>

<sup>6</sup><https://github.com/facebookresearch/faiss>

<sup>7</sup><https://github.com/YiyanXu/DiffRec/>

<sup>8</sup><https://github.com/YangZhengyi98/DreamRec/>

<sup>9</sup><https://github.com/WHUIR/DiffuRec/>

• **LLM2Bert4Rec** (Harte et al., 2023) proposes initializing item embeddings with textual embeddings. In our implementation, we use OpenAI-3-large embeddings, Bert4Rec as backbone and apply PCA to reduce the dimensionality to 64, as mentioned in the original paper.

**Results of Other Backbone.** Here, we present a comparison of PreferDiff with other recommenders using a different backbone, namely GRU. As shown in Table 6, PreferDiff still outperforms DreamRec across all datasets, further validating its versatility. Empirically, we find that, unlike SASRec, which performs better with a Transformer than with GRU4Rec, PreferDiff performs better with GRU as the backbone on the Sports and Toys datasets compared to using a Transformer. This could be due to the relatively shallow Transformer used, making GRU easier to fit. More suitable network architectures for DM-based recommenders will be explored in future work.

Table 6: Comparison of the performance with sequential recommenders with GRU as backbone. The improvement achieved by PreferDiff is significant ( $p$ -value  $\ll 0.05$ ).

Model	Sports and Outdoors				Beauty				Toys and Games			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
GRU4Rec	0.0022	0.0020	0.0030	0.0023	0.0093	0.0078	0.0102	0.0081	0.0097	0.0087	0.0100	0.0090
SASRec	0.0047	0.0036	0.0067	0.0042	0.0138	0.0090	0.0219	0.0116	0.0133	0.0097	0.0170	0.0109
DreamRec	0.0201	0.0147	0.0230	0.0165	0.0431	0.0290	0.0543	0.0321	0.0484	0.0343	0.0591	0.0382
<b>PreferDiff</b>	<b>0.0216</b>	<b>0.0165</b>	<b>0.0250</b>	<b>0.0176</b>	<b>0.0451</b>	<b>0.0313</b>	<b>0.0590</b>	<b>0.0358</b>	<b>0.0530</b>	<b>0.0385</b>	<b>0.0623</b>	<b>0.0415</b>

#### D.4 LEAVE ONE OUT

**Evaluation.** The “leave-one-out” strategy is another widely adopted evaluation protocol in sequential recommendation. For each user’s interaction sequence, the final item serves as the test instance, the penultimate item is reserved for validation, and the remaining preceding interactions are utilized for training. During testing, the ground-truth item of each sequence is ranked against a set of candidate items, allowing for a comprehensive assessment of the model’s ranking capabilities. Performance is evaluated by computing ranking-based metrics over the test set, and the final reported result is the average metric across all users in the test set.

Table 7: Detailed Statistics of Datasets after Preprocessing in Leave-One-Out Setting.

Datasets	Sports	Beauty	Toys	Automotive	Music	Office
#Sequences	35,598	22,363	19,412	2,929	1,430	4,906
#Items	18,357	12,101	11,924	1,863	901	2,421
#Interactions	296,337	198,502	167,597	20,473	10,261	53,258
Avg. Length	8.32	8.87	8.63	6.99	7.17	10.86

**Datasets.** Except for the original three datasets (Sports, Toys and Beauty) in TIGER, we select three additional product review categories—namely, “Automotive”, “Music Instrument” and “Office Product” from Amazon 2014 for a more comprehensive comparison. Here, we utilize the common five-core datasets, filtering out users and items with fewer than five interactions across all datasets.

**Baselines.** Here, we directly report baseline results (e.g.,  $S^3$ -Rec (Zhou et al., 2020), P5 (Geng et al., 2022), FDSA (Hao et al., 2023)) from TIGER (Rajput et al., 2023) and evaluate DreamRec (Yang et al., 2023b) and the proposed PreferDiff.

**Results.** Tables 8 and Tables 9 present the performance of PreferDiff compared with six categories sequential recommenders. For brevity, R stands for Recall, and N stands for NDCG. The top-performing and runner-up results are shown in bold and underlined, respectively. “Improv” represents the relative improvement percentage of PreferDiff over the best baseline. We observe that in the leave-one-out setting, PreferDiff demonstrates competitive recommendation performance compared to the baselines. Specifically, on larger datasets (i.e., Sports and Beauty), PreferDiff performs on par with TIGER. However, on the Toys dataset and the three smaller datasets, PreferDiff achieves a significant lead. This may be due to PreferDiff adopting the same manner as DreamRec, where recommendation is not included in the training process. With a smaller number of items, this approach can result in more precise recommendation performance.

Table 8: Performance comparison on sequential recommendation under leave one out. The last row depicts % improvement with PreferDiff relative to the best baseline.

Methods	Sports and Outdoors				Beauty				Toys and Games			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
P5	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
Caser	0.0116	0.0072	0.0194	0.0097	0.0205	0.0131	0.0347	0.0176	0.0176	0.0166	0.0270	0.0141
HGN	0.0189	0.0120	0.0313	0.0159	0.0325	0.0206	0.0540	0.0257	0.0266	0.0321	0.0497	0.0277
GRU4Rec	0.0129	0.0086	0.0204	0.0111	0.0164	0.0113	0.0283	0.0137	0.0137	0.0097	0.0176	0.0084
BERT4Rec	0.0115	0.0075	0.0191	0.0099	0.0263	0.0184	0.0407	0.0214	0.0170	0.0161	0.0310	0.0183
FDSA	0.0182	0.0128	0.0288	0.0156	0.0261	0.0201	0.0407	0.0228	0.0228	0.0150	0.0381	0.0199
SASRec	0.0233	0.0162	0.0412	0.0209	0.0462	0.0387	0.0605	0.0318	0.0463	0.0463	0.0675	0.0374
S <sup>3</sup> -Rec	0.0251	0.0161	0.0385	0.0204	0.0380	0.0244	0.0647	0.0327	0.0327	0.0294	0.0700	0.0376
DreamRec	0.0087	0.0071	0.0096	0.0075	0.0318	0.0257	0.0624	0.0273	0.0422	0.0347	0.0689	0.0362
TIGER	0.0264	0.0181	0.0400	<b>0.0225</b>	0.0454	<b>0.0321</b>	0.0648	0.0384	0.0521	0.0371	0.0712	0.0432
<b>PreferDiff</b>	<b>0.0275</b>	<b>0.0190</b>	<b>0.0405</b>	0.0218	<b>0.0455</b>	0.0317	<b>0.0660</b>	<b>0.0388</b>	<b>0.0603</b>	<b>0.0403</b>	<b>0.0851</b>	<b>0.0483</b>
<b>Improve</b>	<b>4.16%</b>	<b>4.97%</b>	<b>1.25%</b>	-3.1%	<b>0.22%</b>	-1.25%	<b>1.85%</b>	<b>1.04%</b>	<b>15.73%</b>	<b>8.63%</b>	<b>19.52%</b>	<b>11.81%</b>

Table 9: Performance comparison on sequential recommendation under leave one out. The last row depicts % improvement with PreferDiff relative to the best baseline.

Methods	Automotive				Music				Office			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
DreamRec	0.0543	0.0400	0.0683	0.0445	0.0622	0.0414	0.0783	0.0467	0.0523	0.0378	0.0699	0.0434
TIGER	0.0454	0.0290	0.0745	0.0383	0.0532	0.0358	0.0840	0.0456	0.0462	0.0299	0.0746	0.0390
<b>PreferDiff</b>	0.0649	0.0463	0.0864	0.0532	0.0650	0.0453	0.0874	0.0526	0.0538	0.0379	0.0850	0.0480
<b>Improve</b>	<b>19.52%</b>	<b>15.75%</b>	<b>15.97%</b>	<b>19.55%</b>	<b>4.50%</b>	<b>9.42%</b>	<b>4.04%</b>	<b>12.63%</b>	<b>2.87%</b>	<b>0.26%</b>	<b>13.90%</b>	<b>10.60%</b>

## D.5 GENERAL SEQUENTIAL RECOMMENDATION

**Pretraining Datasets.** Here, we introduce more details about Pretraining datasets. Following the previous work (Hou et al., 2022a; Li et al., 2023a), we select five different product reviews from Amazon 2018 (Ni et al., 2019), namely, “Automotive”, “Cell Phones and Accessories”, “Grocery and Gourmet Food”, “Musical Instruments” and “Tools and Home Improvement”, as pretraining datasets. “Cell Phones and Accessories” is selected as the validation dataset for early stopping when Recall@5 (i.e., R@5) shows no improvement for 20 consecutive epochs. The detailed statistics of each dataset used for pretraining are shown in Table 10. Clearly, the pretraining datasets have no domain overlap with the unseen datasets used in Section 4.2.

Table 10: Detailed Statistics of Pretraining Datasets.

Datasets	Automotive	Phones	Tools	Instruments	Food
#Sequences	193,651	157,212	240,799	27,530	127,496
#Items	18,703	12,839	22,854	2,494	11,778
#Interactions	806,939	544,339	1,173,154	110,151	623,940
Avg. Length	7.26	6.51	7.19	7.06	7.24

**Baselines.** Here, we introduce more details for baselines in General Sequential Recommendation tasks. Notably, for a fair comparison, we employ the text-embedding-3-large model from OpenAI (Neelakantan et al., 2022) as the text encoder instead of Bert (Devlin et al., 2019) in UniSRec and MoRec to convert identical item descriptions (e.g., title, category, brand) into vector representations, as it has been proven to deliver commendable performance in recommendation (Harte et al., 2023). Different of the Mixed-of-Experts (MoE) Whitening utilized in UniSRec, we employ identical ZCA-Whitening (Bell & Sejnowski, 1997) for the textual item embeddings for MoRec and Our proposed PreferDiff.

- **UniSRec** (Hou et al., 2022a) uses textual item embeddings from frozen text encoder and adapts to a new domain using an MoE-enhance adaptor. We adopt the author’s public implementation<sup>10</sup>.
- **MoRec** (Yuan et al., 2023) uses textual item embeddings from frozen text encoder and utilize dimension transformation technique. The architecture is the same as previously mentioned.

<sup>10</sup><https://github.com/RUCAIBox/UniSRec>

**Positive Correlation Between Training Data Scale and General Sequential Recommendation Performance.** Here, we explore how the scale of training data impacts the general sequential recommendation performance of PreferDiff-T. For brevity, we use the initials to represent each dataset. For example, “A” stands for Automotive, and “P” stands for Phones. “AP” indicates that the training data for pretraining includes both Automotive and Phones datasets’ training set.

We observe that both NDCG and HR increase as the training data grows, indicating that PreferDiff-T can effectively learn general knowledge to model user preference distributions through pre-training on diverse datasets and transfer this knowledge to unseen datasets via advanced textual representations. Further studies can explore whether homogeneous datasets lead to greater performance improvements (e.g., whether Amazon Book data provides a larger boost for Goodreads compared to other datasets) and investigate the limits of data scalability for PreferDiff-T.

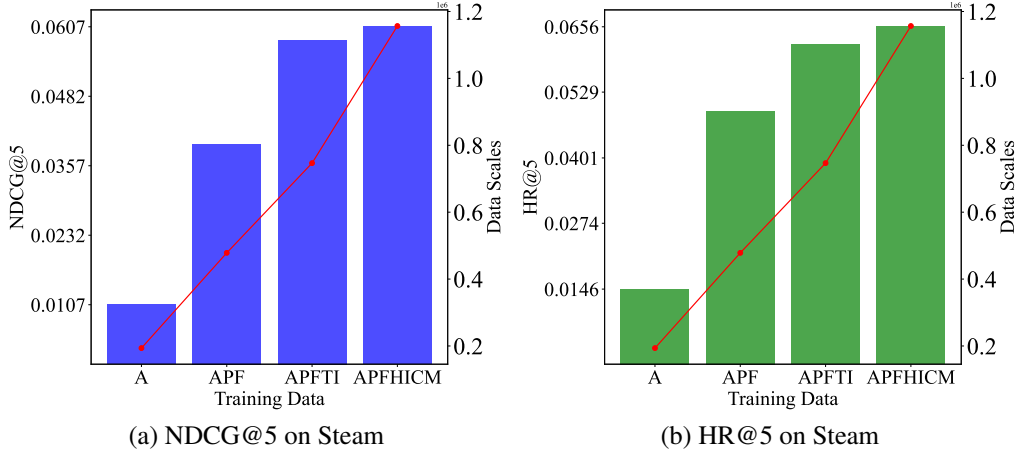


Figure 5: Positive Correlation Between Training Data Scale and General Sequential Recommendation Performance.

## D.6 HYPERPARAMETER SEARCH SPACE

Here, we introduce the hyperparamter search space for baselines and PreferDiff.

Table 11: Hyperparameters Search Space for Baselines.

	Hyperparameter Search Space
GRU4Rec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0
SASRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0
Bert4Rec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, mask probability $\sim \{0.2, 0.4, 0.6, 0.8\}$
CL4SRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, $\lambda \sim \{0.1, 0.3, 0.5, 1.0, 3.0\}$
DiffRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, noise scale $\sim \{1e-1, 1e-2, 1e-3, 1e-4, 1e-5\}$ , $T \sim \{2, 5, 20, 50, 100\}$
DreamRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, embedding size $\sim \{64, 128, 256, 1024, 1536, 3072\}$ , $w \sim \{0, 2, 4, 6, 8, 10\}$
DiffuRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, embedding size $\sim \{64, 128, 256, 1024, 1536, 3072\}$
UniSRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, $\lambda \sim \{0.05, 0.1, 0.3, 0.5, 1.0, 3.0\}$
TIGER	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay $\sim \{0, 1e-1, 1e-2, 1e-3\}$
MoRec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, text-encoder=text-embedding-3-large
LLM2Bert4Rec	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , weight decay=0, text-encoder=text-embedding-3-large
PreferDiff	$lr \sim \{1e-2, 1e-3, 1e-4, 1e-5\}$ , $\lambda \sim \{0.2, 0.4, 0.6, 0.8\}$ , embedding size $\sim \{64, 128, 256, 1024, 1536, 3072\}$ , $w \sim \{0, 2, 4, 6, 8, 10\}$

Table 12: Best Hyperparameters for PreferDiff on Sports, Beauty, and Toys.

Dataset	learning rate	weight decay	$\lambda$	$w$	embedding-size
Sports	1e-4	0	0.4	2	3072
Beauty	1e-4	0	0.8	6	3072
Toys	1e-4	0	0.5	4	3072

## E HYPERPARAMETER ANALYSIS FOR PREFERDIFF

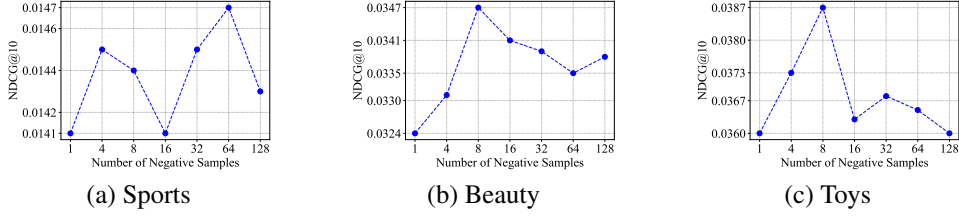


Figure 6: Effect of the Number of Negative Samples for PreferDiff.

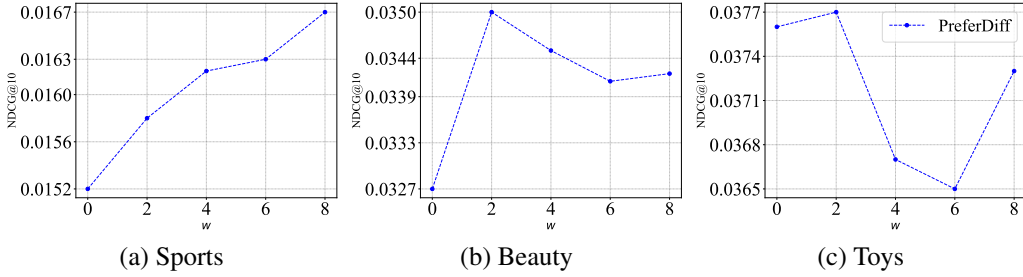


Figure 7: Effect of the  $w$  for PreferDiff.

### E.1 THE NUMBER OF NEGATIVE SAMPLES FOR PREFERDIFF.

Here, we discuss the impact of the number of negative samples on PreferDiff. As shown in Figure 6, we observe that in cases where the number of items is relatively small (e.g., Beauty and Toys), 8 negative samples are sufficient. However, as the number of items increases, the required number of negative samples also grows (e.g., in Sports).

### E.2 IMPORTANCE OF GUIDANCE STRENGTH FOR PREFERDIFF

$w$  controls the weight of personalized guidance during the inference stage of PreferDiff. As shown in Figure 7, increasing  $w$  can enhance recommendation performance. However, an excessively large  $w$  may reduce the generalization capability of DMs, negatively impacting the recommender’s performance. Therefore, we think setting  $w \in [2, 4]$ .

### E.3 DIFFERENT TEXT ENCODERS

Table 13: Comparison of the PreferDiff-T performance with different text-encoder.

PreferDiff-T Text-Encoders	Sports and Outdoors				Beauty				Toys and Games			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Bert	0.0022	0.0020	0.0030	0.0023	0.0104	0.0128	0.0154	0.0148	0.0051	0.0022	0.0068	0.0044
T5	0.0011	0.0009	0.0014	0.0011	0.0241	0.0198	0.0282	0.0212	0.0283	0.0240	0.0309	0.0248
Robert	0.0115	0.0098	0.0135	0.0102	0.0331	0.0256	0.0393	0.0276	0.0391	0.0303	0.0438	0.0319
Mistral-7B	0.0166	0.0130	0.0213	0.0146	0.0375	0.0287	0.0456	0.0312	0.0427	0.0328	0.0505	0.0353
LLaMA-7B	0.0171	0.0126	0.0205	0.0137	0.0402	0.0297	0.0483	0.0323	0.0397	0.0298	0.0494	0.0330
OpenAI-Ada-V2	0.0160	0.0126	0.0183	0.0134	0.0407	0.0318	0.0469	0.0338	0.0396	0.0315	0.0467	0.0339
OpenAI-3-large	<b>0.0182*</b>	<b>0.0145*</b>	<b>0.0222*</b>	<b>0.0158*</b>	<b>0.0429*</b>	<b>0.0327*</b>	<b>0.0532*</b>	<b>0.0360*</b>	<b>0.0460*</b>	<b>0.0351*</b>	<b>0.0525*</b>	<b>0.0387*</b>

**Obtaining Item Embedding from Advanced Text Encoder** Here, we introduce the process for obtaining item embeddings from current advanced text-encoders. For encoder-based large language models, such as Bert (Devlin et al., 2019) and Robert (Liu et al., 2019), we leverage the final hidden state representation associated with the [CLS] token (Hou et al., 2024b). For convenient, we directly utilize the Sentence Transformers APIs<sup>11</sup>. As for other large language models, including T5 (Ni et al.,

<sup>11</sup><https://huggingface.co/sentence-transformers>

2022), Llama-7B (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023), we utilize the output from the last transformer block corresponding to the final input token (Vaswani et al., 2017). Closed-source large language models like text-embedding-ada-v2 and text-embeddings-3-large, we obtain the item embeddings directly via OpenAI APIs<sup>12</sup> (Neelakantan et al., 2022).

**Results.** Table 13 shows the PreferDiff-T employing different item embeddings encoded from text-encoders with varying parameter sizes and architectures. We can observe that

**Positive Correlation Between LLM Size and Recommendation Performance.** The results show that OpenAI-3-large outperforms all other models, indicating that larger language models (LLMs) yield better results in recommendation tasks. This is because larger models generate richer and more semantically stable embeddings, which improve PreferDiff’s ability to capture user preferences. Thus, the larger the LLM, the better the embeddings perform within PreferDiff.

**High-Quality Embeddings Improve Generalization.** Models like Mistral-7B and LLaMA-7B, although smaller than OpenAI-3-large, still perform relatively well across metrics. This suggests that while model size is important, the quality of embeddings plays a crucial role. Especially in the Beauty, these models provide embeddings with sufficient semantic power to enhance recommendation quality.

#### E.4 ANALYSIS OF LEARNED ITEM EMBEDDINGS

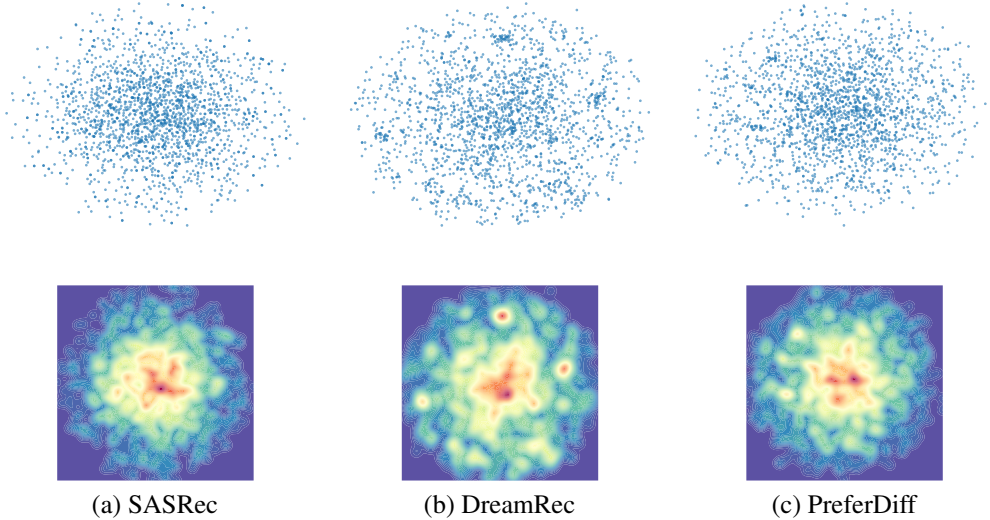


Figure 8: t-SNE Visualization and Gaussian Kernel Density Estimation of Learned Item Embeddings on Amazon Beauty.

To further analysis the item space learned by PreferDiff, we reduce the dimensionality of the learned item embeddings using T-SNE (Van der Maaten & Hinton, 2008)<sup>13</sup> to visualize the underlying distribution of the item space learned by PreferDiff. Due to the large number of items in Amazon Beauty, we randomly select 2000 items as example. Then, we apply Gaussian kernel density estimation (Botev et al., 2010)<sup>14</sup> to analyze the density distribution of reduced item embeddings and visualize the results using contour plots. The red regions indicate areas where a high concentration of items is clustered. From figure 8, we can observe that comparing with SASRec, PreferDiff not only explores the item space more thoroughly (covering most regions). Comparing with DreamRec, PreferDiff exhibits a stronger clustering effect (with high-density regions concentrated in specific areas), better reflecting the similarities between items, result in better recommendation performance.

<sup>12</sup><https://platform.openai.com/docs/guides/embeddings>

<sup>13</sup><https://scikit-learn.org/dev/modules/generated/sklearn.manifold.TSNE.html>

<sup>14</sup>[https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian\\_kde.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html)