FOSTER: First-order Dataset Distillation for Text-based Sequential Recommendation

Anonymous ACL submission

Abstract

Text-based sequential recommender systems, while greatly improving recommendation accuracy by incorporating item contexts, are undeniably more expensive to train. By condensing a large dataset into a compact set of synthetic samples for model training, dataset distillation offers a promising solution. However, its adoption in text-based sequential recommendation is non-trivial given the large pool of discrete items. This challenge is further compounded by language model-based item encoding, which makes bi-level optimization commonly used in dataset distillation prohibitively expensive. To this end, we propose **F**irst-**o**rder data**s**et distillation for **Te**xt-based Sequential **R**ecommendation (**FOSTER**), which facilitates effectiveness and efficiency via three novel components: (1) stochastic item subset sampling that replaces costly full-corpus embedding extraction at each distillation step; (2) first-order optimization with trajectory-anchored parameter reset to avoid expensive bi-level gradient computation; and (3) regularization that explicitly promotes co-occurrence between semantically similar items in the synthetic sequences. Extensive experiments on three benchmarks show that FOSTER consistently outperforms existing dataset distillation and coreset selection baselines, approximating full-dataset performance using as few as 20 synthetic interaction sequences. Code is released at https://anonymous.4open.science/r/foster-7CED.

1 Introduction

Sequential recommendation is the task of suggesting new, unvisited items based on a user's historical interaction sequence. Unlike traditional ID-based methods that directly map item IDs into embeddings for modeling interaction sequences, text-based sequential recommenders leverage language models such as BERT (Yuan et al., 2023) and LLMs (Liu et al., 2024) to encode item textual metadata (e.g., titles and descriptions). With such semantic-aware representations, text-based sequential recommenders benefit from improved generalizability and cold-start robustness, positioning them as a strong direction for next-generation recommender systems (Yuan et al., 2023).

However, these benefits often come with trade-offs in training efficiency, which is vital for recommendation models as they require periodic updates (Lee et al., 2023). This is due to not only the increased data size as each item is now represented by a set of tokens instead of a single ID, but also the overhead from the additional text encoder – commonly a heavy language model – for generating item embeddings. A natural solution is to reduce the training data volume, where existing methods fall into two main categories. Coreset selection methods identify a small, representative subset of the original dataset, preserving real interactions that best capture the underlying data distribution (Lin et al., 2024; Mei et al., 2025; Tran et al., 2026). Dataset distillation, by contrast, generates a compact set of synthetic training samples that compress information learned from the entire dataset, resulting in a more information-dense dataset and superior model performance. These benefits have motivated recent efforts to extend dataset distillation to recommender systems, such as CGM (Wang et al., 2023) and TD3 (Zhang et al., 2025).

Despite the promises, applying dataset distillation to text-based recommendation introduces significant challenges. First, existing methods like TD3 (Zhang et al., 2025) represent each synthetic interaction as a probability over the entire item space, instead of discrete items in real sequences. Though this ensures coverage of all items, replacing discrete items with dense distributions requires computing all item embeddings for every synthetic interaction. The computational cost is further amplified in text-based recommendation, where each

item is represented by a collection of tokens instead of a single ID. This gives rise to the first challenge: **(C1) How to distill quality synthetic sequences from the large pool of items and text tokens?**

Second, dataset distillation is typically formulated as a bi-level optimization problem (Wang et al., 2018a; Zhang et al., 2025) that iteratively refines the synthetic data conditioned on a downstream model trained in the inner loop, therefore incurring high computational costs (Yu et al., 2023). Though recent advances in vision tasks introduce first-order alternatives (Yin et al., 2023; Zhao and Bilen, 2023), these methods are constrained to a small and fixed label space (e.g., object classes) when optimizing synthetic samples. This assumption breaks down in recommendation, where the prediction space corresponds to a large item catalog and each item may serve as a distinct target. Since existing first-order methods optimize separate synthetic samples for each class, their synthetic set size and computational cost scale linearly with the number of items, making them prohibitively expensive and unsuitable for recommendation. Therefore, the second challenge is: **(C2) Can dataset distillation escape the heavy computation trap of bi-level optimization for text-based sequential recommendation?**

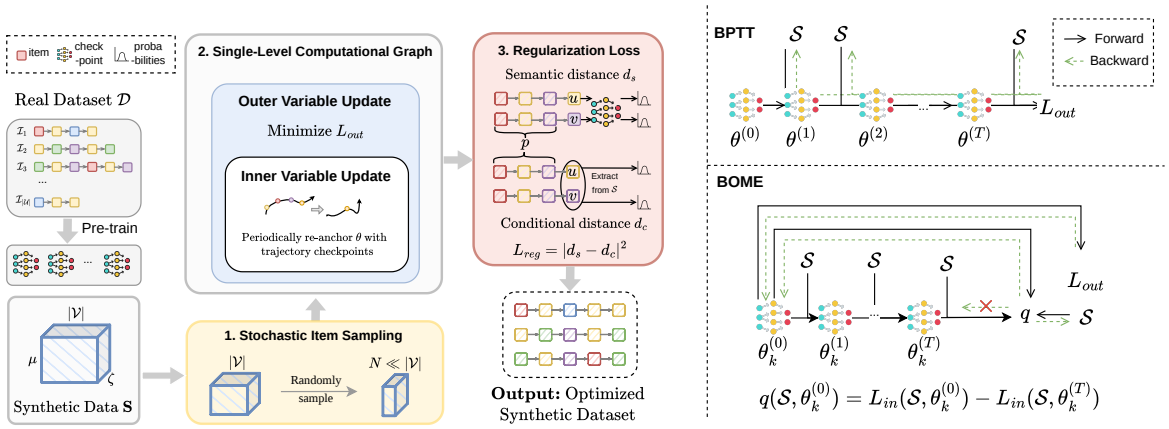
Third, recommender models commonly use tied embeddings, i.e., sharing the same item embedding space for input encoding and output prediction. This practice is theoretically justified only when the underlying data satisfies the distributional hypothesis, where semantically related items tend to appear in similar contexts (Bertolotti and Cazzola, 2024). However, existing recommendation dataset distillation methods overlook the associative patterns among co-occurring items when compress interactions, leading to loss of important semantics that are crucial to recommendation tasks. Thus, the third challenge is: **(C3) How to preserve items' co-occurrence semantics in synthetic sequences to fully capitalize embedding tying?**

Motivated by these challenges, we propose first-order dataset distillation for text-based sequential recommendation (**FOSTER**). Specifically, for **C1**, rather than involving all items at each training step, we instead sample a small random subset of items per batch and compute sequence probabilities directly from their embeddings via Tucker decomposition (Tucker et al., 1964), significantly reducing the per-step computational cost. This is similar with negative sampling: rather than optimizing against the full item catalog at each step as in full softmax, we sample a random subset of items per step. To tackle **C2**, inspired by BOME (Liu et al., 2022), we propose a first-order optimization strategy for sequential recommendation, which jointly optimizes the inner (model weight) and outer (synthetic dataset) parameters. To keep the inner model well-behaved, we employ trajectory-anchored regularization that periodically reverts them to checkpoints sampled from the real training trajectory, preventing drift toward degenerate states unreachable by any realistic training run. Lastly, for **C3**, we introduce a regularization term that encourages the synthetic dataset to satisfy the distributional hypothesis, enabling more reliable dataset results. Figure 1a shows the overview of our method and our contributions are as follows:

1. We formalize the challenges of dataset distillation for text-based sequential recommendation, highlighting the key challenges: (1) expensive full corpus calculation at each timestep, (2) the bi-level trap for text-based sequential recommendation, and (3) the misalignment in preserving the original items' co-occurrence distribution in synthetic datasets.
2. We propose FOSTER, an efficient dataset distillation framework for text-based sequential recommendation. It improves scalability through stochastic item subset sampling, stabilizes first-order bi-level optimization with trajectory-anchored regularization, and preserves item co-occurrence semantics under embedding tying.
3. Extensive experiments on 3 representative datasets, namely: Games, Foods (Ni et al., 2019), and Yelp, show that we can approximate the model performance using only 20 synthetic sequences for Games and Foods, and 60 sequences for Yelp. Furthermore, FOSTER is significantly more efficient and effective than existing recommendation dataset distillation methods.

2 Preliminaries

In this section, we will provide preliminary on text-based sequential recommendation and dataset distillation. After that, we provide the initial pipeline for sequential recommendation introduced by Zhang et al. (2025).



(a) Pipeline Overview

(b) BOME compared to BPTT

Figure 1: (a) Overview of our proposed pipeline. Given real dataset $\mathcal{D}$, we distill it into a compact synthetic dataset $\mathcal{S}$ via stochastic item sampling, a single-level computational graph with inner/outer variable updates, and a regularization loss L_{reg}. (b) Comparison of BOME and BPTT unrolling strategies, illustrating how BOME avoids full trajectory backpropagation by using constraint q.

2.1 Text-based Sequential Recommendation

Given user set $\mathcal{U}$ and item set $\mathcal{V}$, with each item v_i is associated with a textual description d_i, such as title or item description, from which a dense embedding vector e_i is extracted through a language model $l(\cdot, \theta)$, i.e. $\mathbf{e}_i = l(d_i, \theta)$. For each user u, let $\mathcal{I}_u = [v_1, v_2, ..., v_n]$ denotes their chronological ordered interaction sequences. A sequential recommendation model (SeqRec) g_ϕ consumes the sequence of item embeddings $[\mathbf{e}_{v_1}, \ldots, \mathbf{e}_{v_{|\mathcal{I}_u|}}]$ and produces a user representation $\mathbf{h}_u = g_\phi(\mathbf{e}_{v_1}, \ldots, \mathbf{e}_{v_{|\mathcal{I}_u|}})$. At inference, the relevance score between user u and a candidate item v is computed as the dot product $s(u, v) = \mathbf{h}_u^\top \mathbf{e}_v$, and the top-$K$ items are recommended accordingly. Both l_θ and g_ϕ are trained end-to-end by minimizing a binary cross-entropy (BCE) loss over observed and sampled negative interactions, creating the final recommendation model $f = l \cdot g$.

2.2 Dataset Distillation

Given a large training dataset $\mathcal{D} = \{\mathcal{I}_u\}_{u \in \mathcal{U}}$, dataset distillation (Yu et al., 2023; Zhang et al., 2025) aims to synthesize a compact synthetic dataset $\mathcal{S}$ consisting of μ sequences, each of maximum length ζ, such that $|\mathcal{S}| \ll |\mathcal{D}|$, and a model trained on $\mathcal{S}$ achieves performance comparable to one trained on $\mathcal{D}$. Formally, this is defined as a bi-level optimization problem:

$$\begin{aligned} \mathcal{S}^* &= \arg\min_{\mathcal{S}} L_{\text{out}}\left(\theta^*(\mathcal{S}),\, \mathcal{D}\right) \\ \text{s.t. } \theta^*(\mathcal{S}) &= \arg\min_{\theta}\; L_{\text{in}}(\theta,\, \mathcal{S})\,, \end{aligned} \tag{1}$$

where L_{in} is the recommendation loss function of the inner problem that learns SeqRec parameters θ on the synthetic data $\mathcal{S}$. L_{out} is the outer loss function that evaluates θ on the real data $\mathcal{D}$, so as to facilitate the optimization of $\mathcal{S}$.

Since the exact solution $\theta^*(\mathcal{S})$ is intractable, Backpropagation Through Time (BPTT) (Wang et al., 2018a) approximates it by unrolling T steps of gradient descent on the inner objective, and differentiating the outer loss through the entire unrolled computation graph. However, full unrolling with a fixed window suffers from short-horizon bias and unstable gradients. Random Truncated BPTT (RaT-BPTT) (Feng et al., 2024) mitigates this by sampling a random window to train the inner loop, and truncated the last T step only. However, it is still subsumed under a bi-level optimization formulation, inheriting the high computational cost.

2.3 Sequential Recommendation Dataset Distillation

Following (Zhang et al., 2025), we represent the synthetic dataset as a three-dimensional probabilistic tensor $\mathbf{S} \in \mathbb{R}^{\mu \times \zeta \times |\mathcal{V}|}$, where $\mathbf{S}_{ij:} \in \mathbb{R}^{|\mathcal{V}|}$ encodes a soft distribution over the entire item set for synthetic user i at timestep j, allowing gradient-based optimization over the otherwise discrete interaction space.

Directly optimizing $\mathbf{S}$ is intractable when $|\mathcal{V}|$ is large, as the tensor size scales with $\mu \times \zeta \times |\mathcal{V}|$. To address this, $\mathbf{S}$ is parameterized via Tucker decomposition (Tucker et al., 1964):

$$\mathbf{S} = \mathbf{G} \times_1 \mathbf{U} \times_2 \mathbf{T} \times_3 \mathbf{E} =: [\![\mathbf{G}; \mathbf{U}, \mathbf{T}, \mathbf{E}]\!], \tag{2}$$

where $\times_n$ indicates the tensor product along the n-th mode, $\mathbf{U} \in \mathbb{R}^{\mu \times d_1}$ is the synthetic user latent factor, $\mathbf{T} \in \mathbb{R}^{\zeta \times d_2}$ is the temporal dynamics latent factor, $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d_3}$ is the item factor shared with the pretrained item embedding table (frozen during distillation), and $\mathbf{G} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is the relation core with $d_1, d_2, d_3 \ll |\mathcal{V}|$.

In (Zhang et al., 2025), the learnable parameters are optimized with bi-level optimization, where the detailed training objective is discussed in Appendix B. Although the number of learnable parameters reduces drastically, as we mentioned above, this approach requires to compute all item probability in all inner steps, limiting its scalability for text-based sequential recommendation with large item and token sets. In what follows, we present our solution FOSTER that addresses this efficiency challenge while preserving data distillation quality.

3 Methodology

As shown in Fig. 1a, FOSTER consists of three novel components. First, the stochastic item sampling significantly reduces memory usage and computational time at each iteration. Second, a first-order algorithm is designed to bypass the high computational overhead of bi-level optimization. Third, a regularization loss is in place to explicitly enforce distributional hypothesis, preserving items' co-occurrence semantics and uplifting the synthetic data utility. We discuss their details below.

3.1 Item Sampling

A critical scalability bottleneck in Tucker-based distillation in Eq. 2 arises from the item dimension of $\mathbf{S}$, whose size grows linearly with $|\mathcal{V}|$. In the inner loop, all item embeddings must be computed at every step, making distillation on large-catalog datasets prohibitively expensive. To address this, we propose stochastic item subset sampling, which approximates the full-item objective by uniformly drawing a subset of $N \ll |\mathcal{V}|$ items at each step, which we empirically show that suffices to maintain distillation quality while reducing per-step complexity drastically. Formally, at each inner step, we sample a subset $\mathcal{V}_k \subset \mathcal{V}$ of N items and construct a sampled synthetic batch via:

$$\mathbf{S}_k = [\![\mathbf{G}; \mathbf{U}, \mathbf{T}, \mathbf{E}_k]\!], \tag{3}$$

where $\mathbf{E}_k \in \mathbb{R}^{N \times d}$ is the reduced version of the pretrained item embedding table $\mathbf{E}$ of $\mathcal{V}_k$. Crucially, we apply this approximation consistently across both optimization loops.

3.2 First-order Bi-level Optimization for Dataset Distillation

Although the objective in Eq. 1 has wide applicability, it is a bi-level optimization problem and typically requires differentiating through the inner training trajectory, leading to expensive second-order gradients. Inspired by (Liu et al., 2022), we reformulate the bi-level problem as a single-level constrained optimization problem for SeqRec.

Proposition 1 (First-order Optimization)**.** *Suppose $L_{\text{in}}(\mathcal{S}, \theta)$ is continuously differentiable in both arguments, and $\theta^*(\mathcal{S}) \in \arg\min_\theta L_{\text{in}}(\mathcal{S}, \theta)$. We define the constraint*

$$q(\mathcal{S}, \theta) \coloneqq L_{\text{in}}(\mathcal{S}, \theta) - L_{\text{in}}(\mathcal{S}, \theta^*(\mathcal{S})). \tag{4}$$

Then $q(\mathcal{S}, \theta) \geq 0$ with equality iff $\theta \in \arg\min_\theta L_{\text{in}}(\mathcal{S}, \theta)$. The bi-level problem in Eq. 1 is equivalent to

$$\min_{\mathcal{S}, \theta} \; L_{\text{out}}(\theta, \mathcal{S}, \mathcal{D}) \quad s.t. \quad q(\mathcal{S}, \theta) \leq 0, \tag{5}$$

which can be solved by first-order gradients alone.

The proof is provided in Appendix A. In practice, the exact $\theta^*(\mathcal{S})$ is unavailable. We approximate it by θ^T, obtained after T gradient steps on the inner objective, and use

$$\hat{q}(\mathcal{S}, \theta) = L_{\text{in}}(\mathcal{S}, \theta) - L_{\text{in}}(\mathcal{S}, \text{sg}(\theta^T)) \tag{6}$$

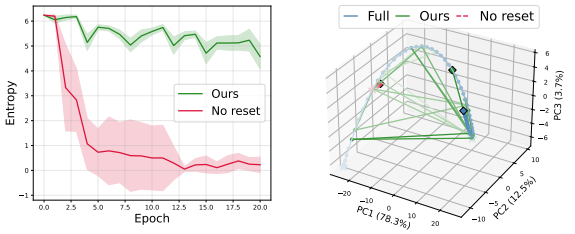
to approximate $q(\mathcal{S}, \theta)$, where $\text{sg}(\cdot)$ denotes stop-gradient. This prevents back-propagation through the construction of θ^T and removes the expensive second-order computation.

To solve Eq. 5, we employ a dynamic barrier update. At the k-th outer step, let $L_k = L_{\text{out}}(\theta_k, S_k, D_k)$ and $\hat{q}_k = \hat{q}(S_k, \theta_k)$. The update direction is

$$\begin{aligned} \delta_k &= \nabla L_k + \lambda_k \nabla \hat{q}_k, \\ \phi_k &= \eta \|\nabla \hat{q}(\mathcal{S}_k, \theta_k)\|^2, \\ \lambda_k &= \max\left(\frac{\phi_k - \langle \nabla L_k, \nabla \hat{q}_k \rangle}{\|\nabla \hat{q}_k\|^2}, \; 0 \right), \end{aligned} \tag{7}$$

where $\eta = 0.5$ controls the barrier strength. We iteratively update $(\mathcal{S}, \theta)$ to reduce L_{out}, while decreasing q to ensure the constraint was held.

However, naively applying such update to dataset distillation leads to degenerate solutions. Since the synthetic dataset $\mathcal{S}$ and the model parameters θ are optimized jointly from scratch, $\mathcal{S}$ tends to overfit to the specific trajectory of θ, resulting



(a) Entropy (b) Training Trajectory

Figure 2: Effect of periodic reset on bilevel distillation. (a) Average sequence entropy over 512 sampled items in distilled sequences. (b) Inner parameters θ in PCA decomposition of the full-data trajectory, our approach and using naive no reset approach.

in a distilled dataset that fails to generalize to independently trained models. To demonstrate this, we calculate the average entropy of the distilled sequences with a fixed 512 synthetic items in Games dataset over the first 20 training epoch. As shown in Fig. 2, $\mathcal{S}$ quickly collapses onto a narrow subset of items, while θ gradually drifts away from the original parameter space.

To address this issue, we avoid maintaining a single θ during optimization. Instead, we anchor the inner variable by sampling from a pre-recorded training trajectory. Let $\mathcal{T} = \{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_M\}$ denote the anchor pool. After R outer loops, we perform a reset step by sampling $\bar{\theta}_k \sim \mathcal{T}$ and setting $\theta_k \leftarrow \bar{\theta}_k$.

Our approach differs from MTT (Cazenavette et al., 2022), which aim to match the original training trajectory, our method aim to improve the performance of the sampled checkpoint. And unlike naive joint optimization, θ_k is reset after a few outer iterations rather than inherited from the previous step. This constrains optimization to valid and diverse model states observed during real training, decouples the distilled data from a single co-evolving model, and encourages $\mathcal{S}$ to capture transferable recommendation patterns.

3.3 Regularization Loss for Distributional Hypothesis Alignment

Recommendation models commonly adopt embedding tying, where the input and output sides share the same item embedding matrix. However, as shown by Bertolotti and Cazzola (2024), this design is theoretically justified only when the distributional hypothesis (DH) holds. However, there is no guarantee that the distillation process naturally produces sequences that satisfy this property. Thus, synthetic data may distort item-level conditional distributions, making semantically similar items conditionally inconsistent. This causes DH to fail, so the tied embedding matrix receives conflicting optimization signals and leads to unstable training and degraded accuracy.

To address this, we introduce a regularization loss L_r that explicitly encourages the synthetic dataset to satisfy the distributional hypothesis. The core idea is to align two notions of item similarity – **semantic equivalent** and **conditional equivalent** – for any randomly sampled pair of items (u, v).

The first is the semantic distance $d_s(u, v)$, where we use the fixed pretrained model f as a proxy to judge whether u and v are semantically similar: if substituting one for the other at the same position in a context p barely changes f's output, the two items are treated as semantically equivalent. Formally, we calculated the semantic distance as:

$$d_s(u, v) = \mathbb{E}_{p\sim\mathcal{S}, i\sim\mathbb{U}(1,\zeta)} |F_i^p(u) - F_i^p(v)| \quad (8)$$

where p and i is a random sequence and position from the synthetic dataset $\mathcal{S}$, respectively; $F_i^p(u)$ is the estimated probability of u given the context $p[:i]$.

The **conditional distance** $d_c(u, v)$ measures how differently u and v are distributed within the synthetic sequences, and can be computed directly from the synthetic data distribution P.

$$d_c(u, v) = \mathbb{E}_{p\sim\mathcal{S}, i\sim\mathbb{U}(1,\zeta)} |P_i^p(u) - P_i^p(v)|, \quad (9)$$

where $P_i^p(u)$ is the conditional probability of u at the i-th position given context p. The regularization loss is then defined as:

$$L_r = \mathbb{E}_{u,v} \left(d_s(u, v) - d_c(u, v)\right)^2, \quad (10)$$

where u and v are uniformly sampled from $\mathcal{V}$. By minimizing L_r, we encourage the semantically similar items to have similar distribution in the synthetic dataset. To instantiate our method for text-based sequential recommendation, we build upon the TD3 framework (Zhang et al., 2025) and add our regularization to the outer-loop objective L:

$$L(\theta, \mathcal{D}, \mathcal{S}) = L_D(\theta, \mathcal{D}) + L_{FM}(\theta, \theta_{\mathcal{D}}) + \lambda_r L_r(\mathcal{S}) \quad (11)$$

where L is the total outer loop loss; $\theta_{\mathcal{D}}$ is the model trained on the real data $\mathcal{D}$. L_D is the binary cross-entropy (BCE) loss, L_{FM} is the feature matching loss as per (Zhang et al., 2025), and L_r and λ_r is the regularization loss and balance weight. The details about L_D, L_{FM} and pseudo-code for the whole process are provided in Appendix B and C, respectively.

Table 1: Pre-processed dataset statistics.

Dataset	#Interactions	#Items	#Users	Avg Length
Games	559,128	35,995	61,639	9.07
Foods	251,085	45,833	25,079	10.01
Yelp	214,722	5,927	17,272	12.43

4 Experiments

In this section, we conduct experiments to evaluate our method. Specifically, we are interested in answering the following research questions (RQs):

RQ1: How does our method compare against other coreset selection and dataset distillation approaches?

RQ2: How does each proposed components contribute to the final performance and method efficiency?

RQ3: Can our generated data be used to effectively and efficiently train other downstream backbones?

RQ4: How sensitive is our method to different hyperparameters?

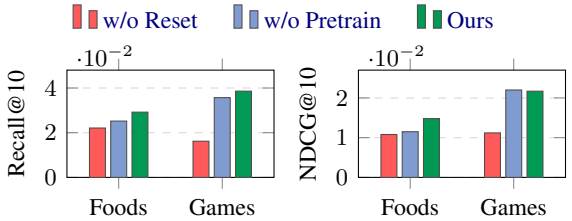
Due to page limit, we defer additional experimental results to the Appendix, including: further hyperparameter sensitivity analysis on numbers of sampled items and synthetic sequences (E.1); further efficiency benchmarking for distillation and downstream training (E.2); and synthetic data visualization (E.4).

4.1 Experiment Settings

We evaluate on three datasets: Amazon Games, Amazon Foods (Ni et al., 2019), and Yelp [1] (filtered to Nashville city). All datasets are split chronologically using an absolute time-based split (Ji et al., 2023) with an 8-1-1 train-validation-test ratio. The pre-processing steps are specified in Appendix D.2. For dataset distillation and coreset selection, we use 20 sequences for Amazon Games and Amazon Foods, and 60 sequences for Yelp. For all datasets, the maximum sequence length is set to 20, with sliding window pre-processing applied. Table 1 show the dataset statistics. The efficiency experiment is conducted on a GPU workstation with an i7-13700K CPU, NVIDIA RTX A5000 GPU, and 32GB RAM.

For the text encoder, we employ a pre-trained TinyBERT (Jiao et al., 2020), and SASRec (Kang and McAuley, 2018) as the recommendation backbone. The model is optimized using AdamW (Loshchilov and Hutter, 2019) with a learning rate of 2e-4. For synthetic and coreset datasets, a weight decay of 1e-4 is applied. The synthetic sequences are optimized with a learning rate of 1e-3 and weight decay of 1e-4. The sampled item set size per step N is set to 512, reset interval R is set to 100, and λ_r is set to 0.1. Appendix D provide more experiment details for baselines and datasets.

[1] https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset



(a) Recall@10 (b) NDCG@10

Figure 3: Ablation study on the Foods and Games datasets. "w/o Reset" removes the parameter reset step; "w/o Pretrain" employs random initialization resets instead of pre-trained trajectory; "Ours" is the full method.

4.2 Main Results (RQ1)

Table 2 shows the main experiment results. Our method achieves the best performance among all approaches and surpasses the full-dataset baseline on several metrics for Games and Foods, demonstrating that our distilled dataset can occasionally exceed full-data training – likely due to the noise-reduction effect, a phenomenon also reported in prior work Lin et al. (2024). Furthermore, our method exhibits a pronounced advantage at smaller K values: gains over the full dataset are more evident at R@10/N@10 than at R@20/N@20. All coreset selection methods perform similarly and fall noticeably short of the Full dataset. Notably, Random selection remains competitive, consistent with observations in Lin et al. (2024); Mei et al. (2025). TD3 achieves a meaningful improvement over coreset methods and approaches full dataset performance.

4.3 Ablation Studies (RQ2)

Reset logic. Figure 3 depicts the ablation study results for inner parameter reset logic. Without resetting the inner parameter, the performance drop drastically, demonstrating the necessity of periodically re-initializing θ. Our proposed approach generally provide a better performance than only using random initialization.

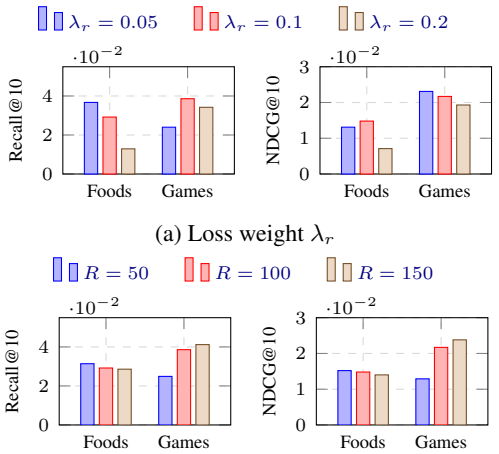
Optimization Logic. Table 3 provides the ab-

Table 2: Main experiment results. "R" and "N" stands for "Recall" and "NDCG", respectively. In each dataset, the best result is marked in **bold** and the second best one is underlined.

Method		Games				Foods				Yelp			
		R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@5	N@5	R@10	N@10
Full		0.0350	0.0178	0.0564	0.0232	0.0228	0.0108	0.0368	0.0143	0.0213	0.0131	0.0390	0.0188
Coreset Selection	Random	0.0317	0.0199	0.0388	0.0217	0.0210	0.0097	0.0295	0.0119	0.0133	0.0086	0.0208	0.0110
	KCenter	0.0311	0.0197	0.0380	0.0215	0.0203	0.0095	0.0292	0.0117	0.0108	0.0093	0.0133	0.0101
	DEALRec	0.0313	0.0199	0.0382	0.0216	0.0202	0.0094	0.0287	0.0115	0.0094	0.0055	0.0152	0.0074
	GORACS	0.0311	0.0195	0.0384	0.0214	0.0204	0.0094	0.0292	0.0116	0.0122	0.0086	0.0176	0.0103
Data Distillation	TD3	0.0338	0.0185	0.0477	0.0219	0.0237	0.0147	0.0356	0.0177	0.0148	0.0093	0.0281	0.0135
	FOSTER	**0.0386**	**0.0217**	**0.0524**	**0.0251**	**0.0292**	**0.0148**	**0.0412**	**0.0178**	**0.0210**	**0.0139**	**0.0340**	**0.0181**

Table 3: Ablation on Games and Foods. We vary the bi-level optimizer (first-order approach or the bi-level BPTT), item selection (on-the-fly (OTF) or K-means (KM)), inner-problem layers (Last or All layers), and the use of regularization L_r. "Time" is the per-epoch runtime (minutes).

First-order	Item	Layers	L_r	Games			Foods		
				R@10	N@10	Time	R@10	N@10	Time
✗	OTF	Last	✓	0.0332	0.0181	9.03	0.0266	0.0126	4.77
✓	OTF	Last	✓	0.0310	0.0202	1.43	0.0252	0.0115	0.58
✓	KM	All	✓	0.0376	0.0222	2.25	0.0278	0.0133	0.98
✓	OTF	All	✗	0.0373	0.0223	2.33	0.0239	0.0114	0.95
✓	OTF	All	✓	0.0386	0.0217	2.42	0.0292	0.0148	1.02



(a) Loss weight λ_r

(b) Reset step R

Figure 4: Hyperparameter sensitive analysis on Foods and Games datasets.

lation study results for various optimization approaches. With the first-order optimizer, when K-means (KM) is in use, instead of using on-the-fly (OTF) sampling, we select K items that are closest to the centroid, and keep this selection fixed through the data distillation process. While OTF incurs a small overhead, its performance is superior as the model can see the whole item set. Also, using all layers for data distillation demonstrates a clear performance advantage. It is worth noting that, when first-order optimizer is turned off in FOSTER, we substitute it with the bi-level BPTT approach. This bi-level approach is evaluated only under the last-layer setting as it requires higher compute resources. As can be seen from the results, FOSTER's efficiency enables it to scale to the all-layer setting, surpassing BPTT in both efficiency and effectiveness.

4.4 Transfer Learning (RQ3)

We conduct transfer learning experiments to evaluate the generalizability of our distilled data across different model architectures. We employ TinyBERT as the backbone for data distillation, then transfer the distilled data to train a larger downstream model, Qwen3-4B, whose detail information is specified in Appendix D.4. Table 4 shows the experiment results. Across all three datasets, our method consistently outperforms Random by a large margin on both R@10 and N@10, while remaining close to full dataset performance. This demonstrates that the distilled data retains sufficient quality to generalize well even when the downstream model differs from the one used during distillation. In terms of training efficiency, our method is considerably faster than full dataset, completing in a fraction of the time.

4.5 Hyperparameter Sensitive (RQ4)

Loss weight λ_r. Figure 4a shows results on Games and Foods datasets with different λ_r. A small $\lambda_r = 0.05$ caused collapse on Games, while perform well on Foods. A large $\lambda_r = 0.2$ demonstrated the opposite results. The middle ground $\lambda_r = 0.1$ seems to provides the best balance, suggesting it effectively regularizes without over-constraining the optimization.

Reset step R. Figure 4b shows that the optimal reset step R varies across datasets. On Foods, smaller values perform better, whereas on Games, larger values yield the highest Recall@10 and NDCG@10. This suggests that different datasets

Table 4: The results of transfer learning, using TinyBERT as backbone for distilling data, while using Qwen3-4B as downstream model. "Time" is the total training time. "h" and "m" stands for "hours" and "minutes", respectively.

Method	Games			Foods			Yelp		
	R@10	N@10	Time	R@10	N@10	Time	R@10	N@10	Time
Full	0.0516	0.0284	21.25h	0.0456	0.0209	13.37h	0.0355	0.0195	5.77h
Ours	0.0449	0.0252	1.55h	0.0435	0.0208	1.23h	0.0314	0.0160	1.33h
Random	0.0229	0.0112	40.7m	0.0129	0.0059	28.5m	0.0207	0.0128	42.87m

require different reset frequencies.

5 Related Work

Text-based Sequential Recommendation. Text-based recommender systems leverage pretrained language models to encode rich semantic information from item textual metadata, overcoming the cold-start limitations of ID-based approaches (Yuan et al., 2023). Notable works include RecFormer (Li et al., 2023), ONCE (Liu et al., 2024), and others (Rajput et al., 2023), which demonstrate that modality-based recommenders are increasingly competitive with, and in some settings surpass, their ID-based counterparts. Our work builds on this paradigm and specifically targets the training efficiency bottleneck that arises when language models are involved in the training loop.

Efficiency for Recommender Systems. For training efficiency, coreset selection methods identify a representative subset of real interactions to reduce training cost, such as DEALRec (Lin et al., 2024), GORACS (Mei et al., 2025), and NaCS (Tran et al., 2026). Dataset distillation goes further by synthesizing compact artificial training samples, with recent works such as ∞-AE (Sachdeva et al., 2022), CGM (Wang et al., 2023), TD3 (Zhang et al., 2025), and DIET (Zhang et al., 2026) exploring this direction in recommendation. However, all of these methods employ expensive bi-level optimization, and only TD3 focused on sequential recommendation settings. There are other works that focuses on recommendation models training efficiency (Qu et al., 2025), but they are orthogonal to our works.

Dataset Distillation. Dataset distillation was first introduced in (Wang et al., 2018b) and has since been extensively studied in image classification, where approaches such as trajectory matching (Cazenavette et al., 2022) and distribution matching (Zhao and Bilen, 2023; Yin et al., 2023) have achieved strong results. In NLP, various dataset distillation methods (Maekawa et al., 2024; Tao et al., 2024; Shen et al., 2025) is also proposed. However, these methods either rely on a small, fixed class space (Zhao and Bilen, 2023; Yin et al., 2023; Shen et al., 2025), an assumption that fundamentally breaks down in recommendation, or rely on an expensive bi-level optimization (Maekawa et al., 2024; Cazenavette et al., 2022; Zhang et al., 2025). For further reading, we refer the reader to (Li et al., 2022).

6 Conclusion

In this paper, we propose FOSTER, namely first-order dataset distillation for text-based sequential recommendation. Our proposed approach includes three main components to further improve the efficiency and the performance. First, instead of using full item set for each timestep, we employ an on-the-fly approach, where we will randomly sample a smaller subset of items. Second, we employ BOME with periodical re-anchor to the inner parameters by the pre-train trajectory checkpoints, drastically reducing the compute resource requirement, while ensuring the synthetic dataset performance. Third, we introduce a regularization loss based on distributional hypothesis to further enhance the final results. Extensive experiments show that our method is not only more efficient but also more accurate than previous state-of-the-art approaches.

Limitations

While our method demonstrates strong empirical results across multiple benchmarks, we identify two limitations that warrant further investigation.

Hyperparameter sensitivity. Our approach introduces several hyperparameters, including the number of inner steps T, the reset interval R, and the number of items sampled each step N. Although we provide sensitivity analyses in Sec. 4.5, selecting these values in practice requires validation effort. Developing more principled or adaptive strategies for hyperparameter selection remains an open direction.

Performance gap and limited LLM transferability. There is still a small performance gap between full dataset and the synthetic dataset in Yelp dataset. Furthermore, while our distilled data transfers effectively to conventional sequential models, the gap between the full dataset and the synthetic dataset for a large language model on the Games and Yelp datasets are still large. We hypothesize this is partly due to the representational gap between the source backbone and the target LLM models, and regard closing this gap as a promising avenue for future work.

References

Francesco Bertolotti and Walter Cazzola. 2024. By tying embeddings you are assuming the distributional hypothesis. In *ICML*, pages 3584–3610.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *CVPR*, pages 10718–10727.

John M Danskin. 2012. *The theory of max-min and its application to weapons allocation problems*. Springer Science & Business Media.

Yunzhen Feng, Shanmukha Ramakrishna Vedantam, and Julia Kempe. 2024. Embarrassingly simple dataset distillation. In *ICLR*.

Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. *TOIS*, 41(3):1–27.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *EMNLP*, pages 4163–4174.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206. IEEE.

Hyunsung Lee, Sungwook Yoo, Dongjun Lee, and Jaekwang Kim. 2023. How important is periodic model update in recommender system? In *SIGIR*, pages 2661–2668.

Guang Li, Bo Zhao, and Tongzhou Wang. 2022. Awesome dataset distillation. https://github.com/Guang000/Awesome-Dataset-Distillation.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In *KDD*, pages 1258–1267.

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024. Data-efficient fine-tuning for llm-based recommendation. In *SIGIR*, pages 365–374.

Bo Liu, Mao Ye, Stephen Wright, Peter Stone, and Qiang Liu. 2022. Bome! bilevel optimization made easy: A simple first-order approach. *NeuRIPS*, 35:17248–17262.

Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *WSDM*, pages 452–461.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Aru Maekawa, Satoshi Kosugi, Kotaro Funakoshi, and Manabu Okumura. 2024. DiLM: Distilling dataset into language model for text-level dataset distillation. In *NAACL*, pages 3138–3153.

Tiehua Mei, Hengrui Chen, Peng Yu, Jiaqing Liang, and Deqing Yang. 2025. Goracs: Group-level optimal transport-guided coreset selection for llm-based recommender systems. In *KDD*, pages 2126–2137.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP*, pages 188–197.

Yunke Qu, Liang Qu, Tong Chen, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2025. Efficient multimodal streaming recommendation via expandable side mixture-of-experts. In *CIKM*, pages 2460–2470.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, and 1 others. 2023. Recommender systems with generative retrieval. *NeuRIPS*, 36:10299–10315.

Noveen Sachdeva, Mehak Dhaliwal, Carole-Jean Wu, and Julian McAuley. 2022. Infinite recommendation networks: A data-centric approach. *NeuRIPS*, 35:31292–31305.

Cheng Shen, Yew-Soon Ong, and Joey Tianyi Zhou. 2025. Condenselm: Llms-driven text dataset condensation via reward matching. In *EMNLP*, pages 1237–1252.

Yefan Tao, Luyang Kong, Andrey Kan, and Laurent Callot. 2024. Textual dataset distillation via language model embedding. In *EMNLP*, pages 12557–12569.

Hung Vinh Tran, Tong Chen, Hechuan Wen, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. 2026. Efficient content-based recommendation model training via noise-aware coreset selection. In *WWW*, pages 6841–6852.

Ledyard R Tucker and 1 others. 1964. The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119:110–182.

Cheng Wang, Jiacheng Sun, Zhenhua Dong, Ruixuan Li, and Rui Zhang. 2023. Gradient matching for categorical data distillation in ctr prediction. In *RecSys*, pages 161–170.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018a. Dataset distillation. *arXiv preprint arXiv:1811.10959*.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. 2018b. Dataset distillation. *arXiv preprint arXiv:1811.10959*.

Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a more user-friendly and easy-to-use benchmark library for recommender systems. In *SIGIR*, pages 2837–2847. ACM.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zeyuan Yin, Eric Xing, and Zhiqiang Shen. 2023. Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. *NeuRIPS*, 36:73582–73603.

Ruonan Yu, Songhua Liu, and Xinchao Wang. 2023. Dataset distillation: A comprehensive review. *TPAMI*, 46(1):150–170.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *SIGIR*, pages 2639–2649.

Jiaqing Zhang, Hao Wang, Mingjia Yin, Bo Chen, Qinglin Jia, Rui Zhou, Ruiming Tang, ChaoYi Ma, and Enhong Chen. 2026. Diet: Learning to distill dataset continually for recommender systems. *arXiv preprint arXiv:2603.24958*.

Jiaqing Zhang, Mingjia Yin, Hao Wang, Yawen Li, Yuyang Ye, Xingyu Lou, Junping Du, and Enhong Chen. 2025. Td3: Tucker decomposition based dataset distillation method for sequential recommendation. In *WebConf 2025*, pages 3994–4003.

Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *WACV*, pages 6514–6523.

A Theoretical Details

Let $L_{\text{in}}(\mathcal{S}, \theta)$ denote the inner loss on the synthetic dataset S with model parameter θ. The inner condition $\theta = \theta^*(\mathcal{S}) \in \arg\min_{\theta'} L_{\text{in}}(\mathcal{S}, \theta')$ enforces θ to be optimal on S. Let

$$L^*_{\text{in}}(\mathcal{S}) = \min_{\theta'} L_{\text{in}}(\mathcal{S}, \theta') \tag{12}$$

be the minimum achievable inner loss. Since $L_{\text{in}}(\mathcal{S}, \theta) \geq L^*_{\text{in}}(\mathcal{S})$ for any θ, the inner optimality condition can be written as the constraint

$$\min_{\mathcal{S},\theta} \; L_{\text{out}}(\theta, \mathcal{S}, \mathcal{D}) \quad \text{s.t.} \quad q(\mathcal{S}, \theta) \leq 0, \tag{13}$$

Because $q(\mathcal{S}, \theta)$ is always non-negative, this constraint is satisfied only when θ reaches the inner optimum on S.

To optimize Eq. (13), we need the constraint gradient

$$\nabla_S q(\mathcal{S}, \theta) = \nabla_1 L_{\text{in}}(\mathcal{S}, \theta) - \nabla_S L^*_{\text{in}}(\mathcal{S}), \tag{14}$$

where the second term depends on the inner optimum $\theta^*(\mathcal{S})$. Directly computing this term would seem to require the implicit gradient $\nabla_S \theta^*(\mathcal{S})$. Next, we will show that this dependence vanishes under the inner first-order optimality condition. By the chain rule,

$$\begin{aligned} \nabla_S L^*_{\text{in}}(\mathcal{S}) = \nabla_1 L_{\text{in}}(\mathcal{S}, \theta^*(\mathcal{S})) \\ + J_{\theta^*}(\mathcal{S})^\top \nabla_2 L_{\text{in}}(\mathcal{S}, \theta^*(\mathcal{S})), \end{aligned} \tag{15}$$

where $J_{\theta^*}(\mathcal{S})$ is the Jacobian of $\theta^*(\mathcal{S})$ with respect to S. Since $\theta^*(\mathcal{S})$ is an inner optimum, the first-order optimality condition gives

$$\nabla_2 L_{\text{in}}(\mathcal{S}, \theta^*(\mathcal{S})) = 0.$$

Therefore, the second term vanishes, yielding

$$\nabla_S L^*_{\text{in}}(\mathcal{S}) = \nabla_1 L_{\text{in}}(\mathcal{S}, \theta^*(\mathcal{S})).$$

Thus, the value-function gradient does not require differentiating through $\theta^*(\mathcal{S})$, which is also a direct consequence of Danskin's theorem (Danskin, 2012). Therefore, Eq. 13 can be solved by first-order gradients alone.

B Sequential Recommendation Dataset Distillation Optimization Objective

The distillation is framed as a bi-level optimization problem where the inner loop trains the sequential encoder on the synthetic summary and the outer loop evaluates it on the real data. Concretely, the inner objective employs KL divergence between the model's predicted distribution and the synthetic soft targets, while the outer loop will optimize the actual objective – improving the synthetic data $\mathbf{S}$. In this section, we will introduce about the optimization process of outer and inner loops.

B.1 Outer Loop

Following standard practice in sequential recommendation (Kang and McAuley, 2018), we employ a binary cross-entropy formulation $L_{\mathcal{D}}$ over autoregressive next-item predictions as the recommendation model's training objective, given by:

$$L_{\mathcal{D}} = \sum_{i=1}^{N} \big(BCE(\langle \hat{\mathbf{e}}_i, \mathbf{e}_i \rangle, 1) + BCE(\langle \hat{\mathbf{e}}_i, \mathbf{e}_i^{neg} \rangle, 0) \big), \tag{16}$$

where $\hat{\mathbf{e}}_i$ is the contextual representation produced at position i used to predict the next item, $\langle \cdot, \cdot \rangle$ is dot product, $\mathbf{e}_i^{neg}$ is the negatively sampled samples at i-th timestep.

Zhang et al. (2025) extends the outer loop objective beyond naive performance matching by incorporating a feature space alignment term:

$$\begin{aligned} \mathbf{S}^* &= \arg\min_{\mathbf{S}} \ \mathcal{L}_{\mathcal{D}}(\theta^*, \mathcal{D}) + L_{FM} \\ L_{FM} &= \frac{1}{2} \left\| f(\theta_{\mathcal{D}}, \mathcal{D}) - f(\theta^*, \mathcal{D}) \right\|_2^2, \\ \text{s.t. } \theta^* &= \arg\min_{\theta} \mathcal{L}_{\text{in}}(\theta, \mathbf{S}) \end{aligned} \tag{17}$$

where L_{FM} is feature matching loss, $f(\theta_{\mathcal{D}}, \cdot)$ and $f(\theta^*, \cdot)$ are the recommendation models trained on the real and synthetic data, respectively. This alignment loss encourages the synthetic-data-trained model to converge to a similar feature space solution as the real-data-trained model, promoting a more favorable loss landscape. The bilevel problem is solved via RaT-BPTT.

B.2 Inner Loop

In the inner loop, we first define:

$$\mathbf{X}_i = \mathbf{S}[i, : \zeta, :], \mathbf{y}_i = \mathbf{S}[i, \zeta], \tag{18}$$

where $\mathbf{S}[i, :, : \zeta, :]$ follows NumPy slice notation, $\mathbf{X}_i \in \mathbb{R}^{(\zeta-1) \times |\mathcal{V}|}$ is the input, $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{V}|}$ denotes the synthetic target distribution. As $\mathbf{X}_i$ represents a probability distribution over items rather than a discrete item sequence, it cannot be directly processed by the recommendation model f. We therefore compute the embedding of $\mathbf{X}_i$ as a weighted average of all (in TD3 (Zhang et al., 2025)) or sampled (in FOSTER) item embeddings $\mathbf{E}$, with weights given by $\mathbf{X}_i$:

$$\mathbf{E}'_i = l(\mathbf{X}_i) = \langle \mathbf{X}_i, \mathbf{E} \rangle, \tag{19}$$

where l is the language model, $\mathbf{E}$ is the embedding of all (sampled) items. Thus, we can process $\mathbf{E}'_i$ normally by the recommendation model g, producing output representation $\mathbf{h}_i = g(\mathbf{E}'_i) = f(\mathbf{X}_i)$. Then, the final output:

$$\hat{\mathbf{y}} = \mathbf{h}_i^\top \mathbf{E}'_i \tag{20}$$

Finally, we employ KL Divergence instead of BCE as the loss function for the inner training loop:

$$L_{in}(\theta, \mathcal{S}) = D_{KL}(\mathbf{y}_i \| \hat{\mathbf{y}}_\mathbf{i}). \tag{21}$$

Using KL divergence is natural in this setting because each synthetic position is not assigned to a single discrete item; instead, it represents a soft supervisory signal that may place non-trivial probability mass on multiple semantically related items.

Algorithm 1 shows this process. We first sampled a smaller item set $\mathcal{V}_k$ (line **1**). Then, we get the embedding $\mathbf{E}_k$ from $\mathcal{V}_k$ (line **2**), note that $\mathbf{E}_k$ is a pre-computed embedding table using a pre-trained model, not the embedding table extracted from the currently in-training l. Then, we sample the input $\mathbf{X}_i$ and the target $\mathbf{y}_i$ (Line **3**). Then, we perform standard pipeline of forward (line **4-5**), calculating loss (line **6**) and update model (line **7**).

Algorithm 1: A training step on the synthetic dataset

Data: Synthetic data tensors $\mathbf{G}, \mathbf{U}, \mathbf{T}$, embedding table $\mathbf{E}$, learning rate α, Real item list $\mathcal{V}$

1 $\mathcal{V}_k \leftarrow$ Random sample from $\mathcal{V}$;
2 $\mathbf{E}'_k \leftarrow$ Get embedding corresponding for $\mathcal{V}_k$;
3 $\mathbf{X}_i, \mathbf{y}_i \leftarrow [\![\mathbf{G}, \mathbf{U}[i], \mathbf{T}, \mathbf{E}_k]\!]$;
4 $\mathbf{E}'_k \leftarrow l(\mathcal{V}_k)$;
5 $\hat{\mathbf{y}}_k \leftarrow g(\mathbf{S} \odot \mathbf{E}'_k)$;
6 $L \leftarrow$ Eq. 21;
7 $g, l \leftarrow g - \alpha \nabla_g L, l - \alpha \nabla_l L$;

C Pseudo-code

Algorithm 2 shows the pseudo-code for the proposed algorithm. First, we need to train the model on the real dataset (Line **1**). This prepare step is common in various dataset distillation approaches, such as MTT (Cazenavette et al., 2022) and TD3 (Zhang et al., 2025). Then, we encode item embedding (Line **2**) to get the item embedding used in calculating $\mathbf{S}$. Then, we can start to distill the synthetic dataset. At each iteration, we first reset θ every R steps from pre-trained trajectory; (Line

Algorithm 2: The proposed dataset distillation algorithm (FOSTER)

Data: Real data $\mathcal{D}$, learning rate α, inner parameter reset intervals R, number of inner loop steps T

```
# Prepare step
```

1 Train θ on $\mathcal{D}$ to get pre-train trajectory;

2 $\mathbf{E} \leftarrow$ Encode all real items $\mathcal{V}_{\mathcal{D}}$;

```
# Main dataset distillation step
```

3 Initialize $\mathcal{S}$ $(\{\mathbf{T}, \mathbf{G}, \mathbf{U}\})$;

4 Iteration $i \leftarrow 0$;

5 **repeat**

6   **if** $i \mod R = 0$ **then**

7         $\theta \leftarrow$ Sample from the pre-train trajectory;

```
# First-order update (Sec. 3.2)
```

8     $\theta^* \leftarrow$ Train θ on $\mathcal{S}$ for T steps;

9     $q(\mathcal{S}, \theta) \leftarrow L_S(\theta, \mathcal{S}) - L_S(\theta^*, \mathcal{S})$;

10     $\lambda_k \leftarrow$ Eq. 7;

11     $L \leftarrow$ Eq. 11;

12     $(\mathcal{S}, \theta) \leftarrow (\mathcal{S}, \theta) - \alpha(\nabla L + \lambda_k \nabla q)$;

```
# End first-order update
```

13     $i \leftarrow i + 1$;

14 **until** *converged*;

6-7). Then, we train θ on $\mathcal{S}$ for T steps to estimate $\theta_{\mathcal{S}}$ (Line **8**), detailed in Alg. 1. Note that in these T steps, we will sample a different item set for each step. After that, we compute constraint $q(\mathcal{S}, \theta)$ (Line **9**); and update $(\mathcal{S}, \theta)$ via the first-order gradient step (Line **10-12**). The process is iterated until convergence.

D Detailed Experiment Settings

D.1 Baselines

In this section, we provide details on baselines used in the main experiments.

- **Random** randomly samples a subset of users and uses their interaction as the coreset.
- **K-Center** represents each user by their last-timestep embedding and applies the greedy k-center algorithm to select the subset.
- **DEALRec** (Lin et al., 2024) scores each training sample using an effort score derived from a pre-trained model, pruning low-effort and redundant sequences.
- **GORACS** (Mei et al., 2025) selects coresets by bounding the test loss via optimal transport distance and gradient information, avoiding repeated model retraining during selection.
- **TD3** (Zhang et al., 2025) generates a compact set of pseudo-sequences via Tucker decomposition to match the distributional statistics of the full training set. For TD3, due to the high computational cost, we only unfreeze the last layer during the inner loop training.

D.2 Data

We source all datasets from RecBole (Xu et al., 2023) and apply standard pre-processing steps. Interaction records are first sorted chronologically by timestamp, then filtered using the 5-core setting (retaining only users and items with at least 5 interactions). For Yelp, we additionally restrict items to those located in Nashville city prior to applying the 5-core filter. During training, we apply a sliding window strategy to generate additional samples from long interaction sequences (Mei et al., 2025), while at evaluation time each sequence is truncated to the last 20 interactions.

D.3 Implementation Details

The inner steps T is searched in $\{40, 50, 60\}$. The model and synthetic data is selected with the best results in validation dataset in R@10. We calculate metrics using all items.

D.4 LLM Implementation Details

Follow (Liu et al., 2024), we employ LLM (Qwen3-4B (Yang et al., 2025)) directly as a text encoder. We apply LoRA with rank 32 and alpha 128, yielding a scaling factor of 4 to the last two transformer layers (layers 34 and 35), with dropout of 0.1. All of the other layers are frozen. To speed up the training process, we adopt a caching strategy wherein we pre-compute and store the hidden states from the first layers for all item embeddings.

E Additional Experiment Results

E.1 Further Hyperparameter Analysis

Number of sequences. Figure 5 shows the experiment results on how the number of sequences affects the downstream model performance.

Number of sampled items. Figure 6 shows the experiment results on how the number of sampled items per step N affects the downstream model performance.

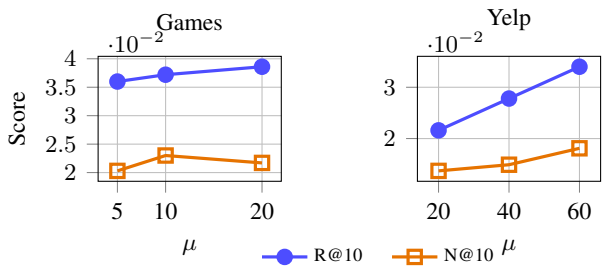


Figure 5: Model performance under varying numbers of distilled sequences μ on Games and Yelp datasets.

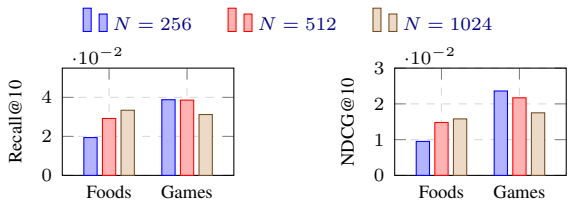


Figure 6: The effect of number of item sampled per step K on the performance.

Table 5: Runtime (minutes) and peak GPU memory usage (MB) per epoch on the Foods dataset, measured on a single NVIDIA RTX A5000 (24GB VRAM). OOM indicates out-of-memory.

Method	#Items	Layers	Time	Mem
TD3	All	Last	OOM	33,002
FOSTER	All	Last	10.43	20,654
TD3	512	Last	2.20	5,004
FOSTER	512	Last	0.58	1,296
TD3	512	All	5.07	19,368
FOSTER	512	All	0.98	5,198

Table 6: Training time comparison between full training and the synthetic data on Foods dataset. "h" and "m" stands for hours and minutes, respectively.

Backbone	Full			Ours		
	R@10	N@10	Time	R@10	N@10	Time
TinyBERT	0.0228	0.0108	22.2m	0.0292	0.0148	3.7m
Qwen3-0.6B	0.0292	0.0148	7.5h	0.0368	0.0179	54.2m
Qwen3-4B	0.0391	0.0182	13.4h	0.0435	0.0208	1.2h

E.2 Efficiency Benchmark

Table 5 reports per-epoch runtime and peak GPU memory on Foods. TD3 (Zhang et al., 2025) immediately exceeds the 24GB VRAM budget, confirming that naive bi-level optimization does not scale to large catalogues. Reducing to a 512-item subset via stochastic sampling yields the largest gains: FOSTER requires only 1,296MB and 0.58 minutes per epoch — a $25.46\times$ memory reduction over TD3. Even under all-layer fine-tuning, FOSTER (5,198MB, 0.98 minutes) remains substantially cheaper than TD3 (19,368MB, 5.07 minutes), demonstrating that first-order optimization is critical for tractable training under consumer-grade GPU constraints.

Table 6 shows the training efficiency on Foods dataset. Our method successfully not only approximate the performance of the full training, but also drastically reduce the required training time.

E.3 Further results analysis

In this section, we further analyze the lower results on Yelp. Figure 7 illustrates the normalized item popularity distributions across three datasets. All three datasets exhibit a power-law decay, where a small fraction of items account for the majority of interactions. However, the datasets differ substantially in catalog size and tail behavior. Both Foods and Games maintain extended long tails spanning over 30,000 items. In contrast, Yelp has a substantially smaller catalog (5,000 items) with a steeper decay and an abrupt tail cutoff. Because Yelp's naturally short and sparse tail provides almost no gradient signal for tail items during optimization process, the synthetic sequences collapse toward the popular items.

E.4 Data Visualization

We calculate the probabilities of all items, and for each user, we visualized the top 10 items that has highest probabilities.

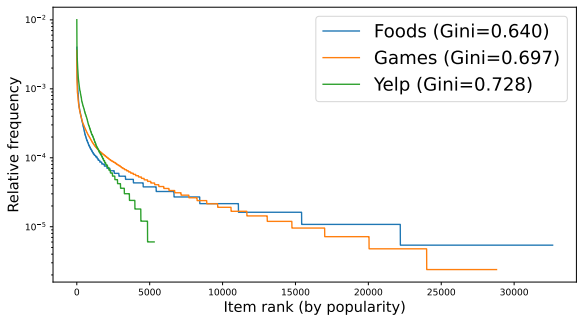


Figure 7: Item popularity distributions of the three training datasets.

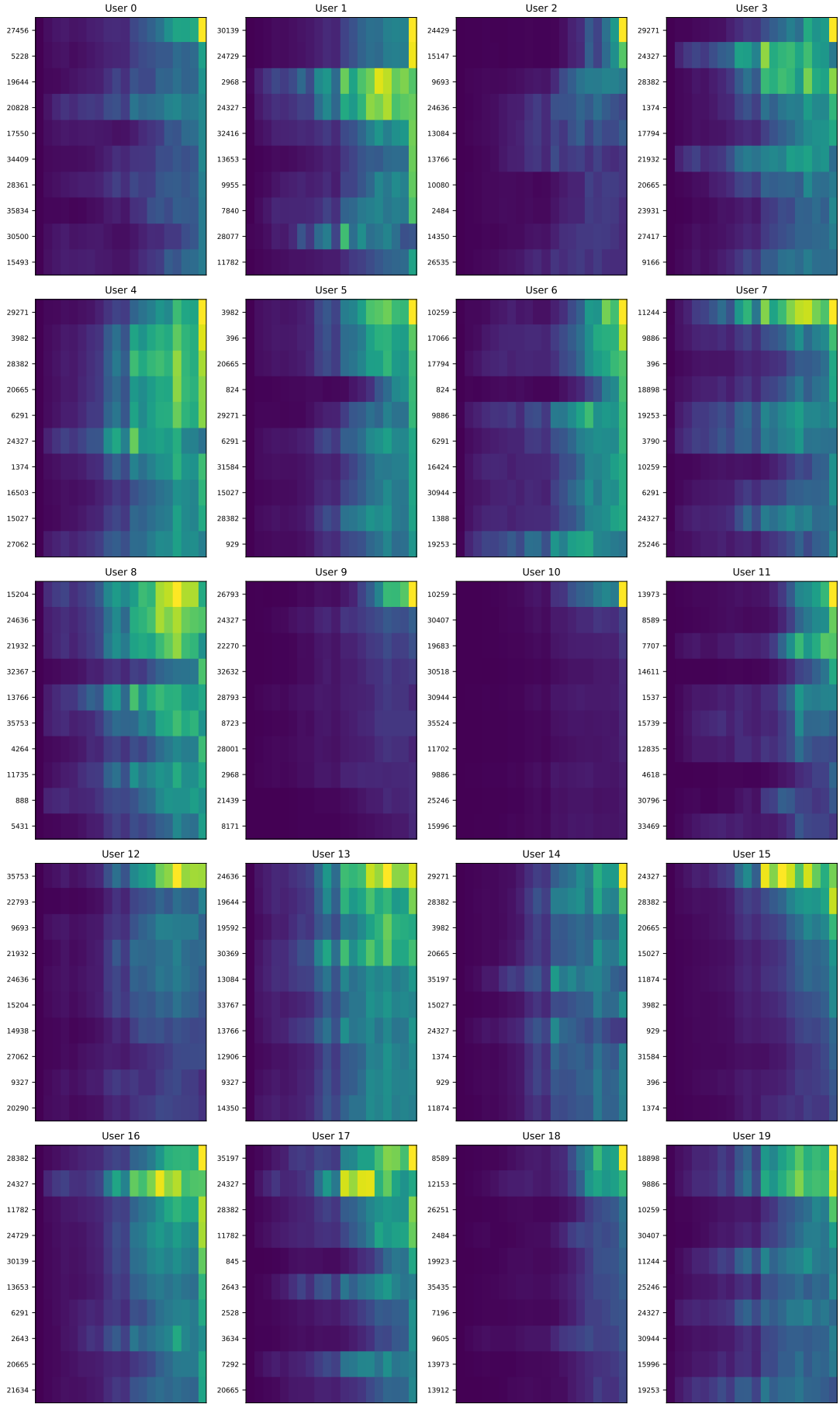


Figure 8: Games dataset visualization

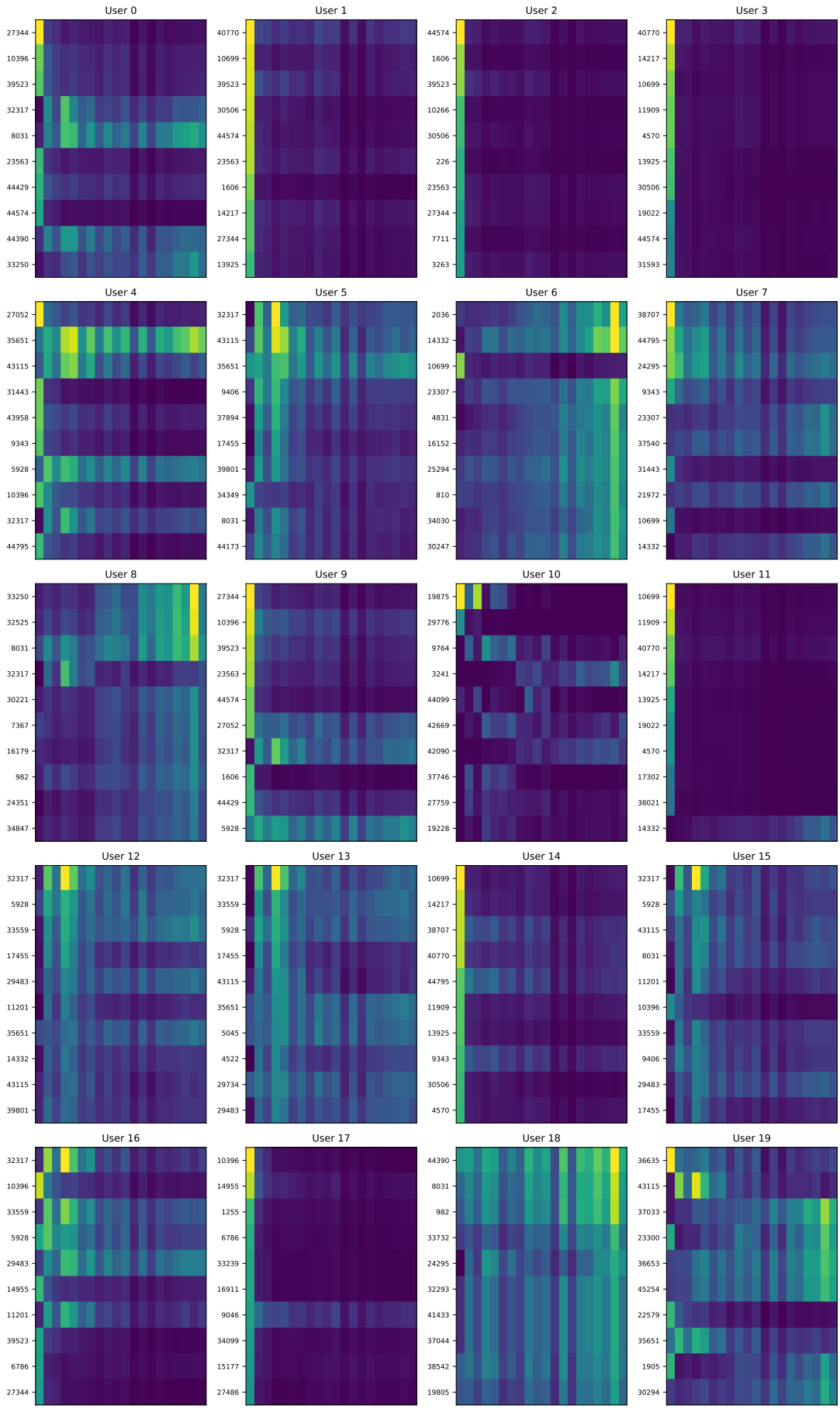


Figure 9: Foods dataset visualization