DART: Diversified and Accurate Long-Tail Recommendation

Jeongin Yun, Jaeri Lee, and U Kang*

Seoul National University, Seoul, South Korea
{yji00828, jlunits2, ukang}@snu.ac.kr

Abstract. *How can we accurately recommend unpopular items and increase their exposure to users?* Previous models fail to handle the skewed distribution of item interactions, resulting in an overemphasis on popular items and inadequate recommendations of unpopular tail items. Existing approaches that address this imbalance often sacrifice overall accuracy to boost the accuracy of recommending tail items, or overlook the limited presence of tail items in the recommendations. In this paper, we propose DART (Diversified and AccuRate Long-Tail Recommendation), an accurate and diversified recommendation method which evenly recommends items across all popularity groups while maintaining high accuracy within each group. We increase the interactions of tail items by generating synthetic sequences which preserve original user preferences. DART improves the representation of tail items through contrastive learning which facilitates the learning of the relationships between similar head and tail items. Additionally, we ensure that only reliable information is learned in the embedding of tail items through a popularity-based negative sampling. Experimental results demonstrate that DART achieves up to 44.7% higher Coverage@10 and 47.5% higher nDCG@10 for tail items as ground-truth compared to the best competitor while improving the overall nDCG@10 by up to 22%.

Keywords: Sequential Recommendation · Diversified Recommendation · Long-tail Distribution · Contrastive Learning

1 Introduction

How can we accurately recommend unpopular items and increase their exposure to users? Recommending less popular items is essential not only for balanced inventory management but also for helping users to make serendipitous discoveries. However, this task is challenging owing to the long-tail distribution of real-world transaction datasets. A few head items dominate purchases, leaving the majority of items as tail items with significantly fewer interactions, as shown in Fig. 1a. Models that fail to account for this distribution predominantly recommend popular items while rarely exposing tail items. A representative recommendation model, SASRec [9], rarely recommends unpopular items while excessively recommending popular items, as shown in Fig. 1c. Moreover, Fig. 1b

* Corresponding author

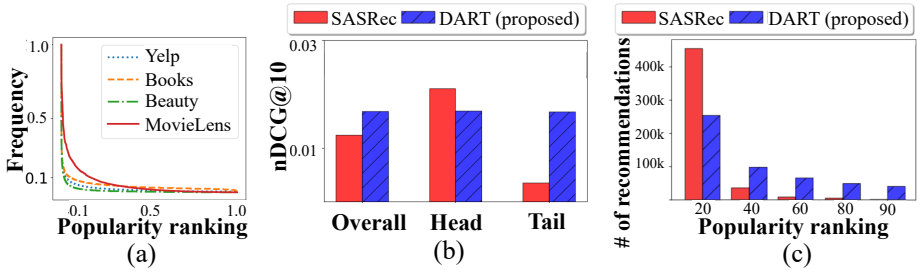

Fig. 1: (a) Long-tail distributions of real-world datasets. Values on each axis are normalized. (b) Performance on overall, head, and tail items. (c) Recommendation frequency according to item popularity.

demonstrates that SASRec exhibits poor accuracy when the ground-truth items are tail items with a significant performance gap compared to popular items.

The challenges in recommending tail items arise from insufficient and distorted training of their embeddings. The inherently infrequent occurrence of tail items limits their learning opportunities, as their embeddings are trained only when interactions with those items occur. Recent efforts to address this issue merely supplement the insufficient information of tail items with additional content (such as item text or category) [1, 8, 17, 22] or data from head items [13, 18]. Additionally, tail items are susceptible to distortion from random negative sampling, a common technique in recommendation models. Owing to the limited training data available for tail items, even a single weight update from inaccurate negative examples critically impacts the model's performance on them. However, existing tail recommendation models [13, 26] employ random negative sampling without addressing these effects on tail items.

In this paper, we propose DART (Diversified and AccuRate Long-Tail Recommendation), an accurate and diversified sequential recommendation model. DART evenly recommends both head and tail items as shown in Fig. 1c, and achieves high accuracy for both types of items in Fig. 1b. To handle the skewed distribution, DART generates synthetic sequences that increase the occurrence of tail items; these sequences maintain the preferences of base sequences by finding similarity relationships between head and tail items. Additionally, DART precisely trains the embeddings of tail items via contrastive learning, which enables the learning of the similarity relationship. DART ensures undistorted training of tail item embeddings via popularity-based negative sampling.

Our contributions are summarized as follows:

- **Diversified tail recommendation.** DART accurately recommends a diverse range of tail items while also enhancing the overall recommendation accuracy, unlike existing methods, which focus only on tail or overall accuracy.
- **Specialized contrastive learning.** We design a contrastive learning approach aimed at the accurate training of tail items, ensuring the transfer of the proposed clustering information to item embeddings.
- **Performance.** We show that DART achieves up to 44.7% higher Tail Coverage@10 and 47.5% higher Tail nDCG@10 compared to the best competitor. The code is available at `https://github.com/snudatalab/DART`.

2 Preliminaries and Related Works

2.1 Sequential Tail Recommendation

Problem definition. Let $u \in \mathcal{U}$ and $i \in \mathcal{I}$ denote a user and item, respectively. Given a set $\mathcal{S}$ of sequences, where each sequence $\mathbf{s}_u = (i_1, i_2, ..., i_m)$ of user u is chronologically ordered with length m, and $i_m \in \mathcal{I}$ denotes the item purchased at timestamp m, the task is to recommend to each user u a ranked list of top-K items that the user will interact with at timestamp $(m+1)$. The goal is to maximize 1) tail accuracy (accuracy when the ground-truth item is a tail item) and 2) tail coverage (proportion of recommended tail items for all users to the tail item set) without compromising the overall accuracy.

Distinguishing between head and tail items. Items are sorted by purchase frequency in descending order, with the bottom $\beta\%$ defined as the tail item set $\mathcal{I}^\tau$ and the remainder as the head item set $\mathcal{I}^\eta$. We set β to 80 according to the well-known Pareto Principle [20]. Note that $\mathcal{I}^\tau \cap \mathcal{I}^\eta = \varnothing$ and $\mathcal{I}^\tau \cup \mathcal{I}^\eta = \mathcal{I}$.

2.2 Sequential Recommendation Framework

General sequential recommendation models [6, 12, 14, 15] encode a sequence into a representation vector. The sequence representation vector $\mathbf{e}_{\mathbf{s}_u} = f_\theta(\mathbf{s}_u)$ is generated by encoding the sequence $\mathbf{s}_u$ of user u with the sequence encoder $f_\theta(\cdot)$. The matching score $r_{u,i} = \mathbf{e}_{\mathbf{s}_u}^\top \mathbf{e}_i$ between user u and item i is computed by the dot product of $\mathbf{e}_u$ and the embedding vector $\mathbf{e}_i$ of item i.

2.3 Tail Recommendation

Long-tail distribution is critical in recommendation systems as recommendations influence users' decisions and exacerbate skewness [5, 11]. Approaches to address this include leveraging head items [13, 18], incorporating external data [19], and using sequence augmentation [25]. CoLTRec [18] and MELT [13] infer tail item information from head items. LOAM [25] generates synthetic sequences by weighting tail items in item graphs. The proposed approach enhances both overall and tail accuracy while also improving tail coverage, whereas previous studies typically focus on enhancing one of them. We achieve this using self-supervised signals from user interactions without additional data.

2.4 Contrastive Learning

Contrastive learning [7] enhances representations by pulling similar samples closer and pushing unrelated ones apart. It has recently been applied to sequential recommendation systems to refine sequence representations. CL4SRec [24] learns robust representations via augmented sequences, ICLRec [2] captures latent user intent, and DCRec [26] mitigates popularity bias. In contrast, we address the long-tail problem by proposing a positive pairing strategy that incorporates cluster information to capture the relationships between items.

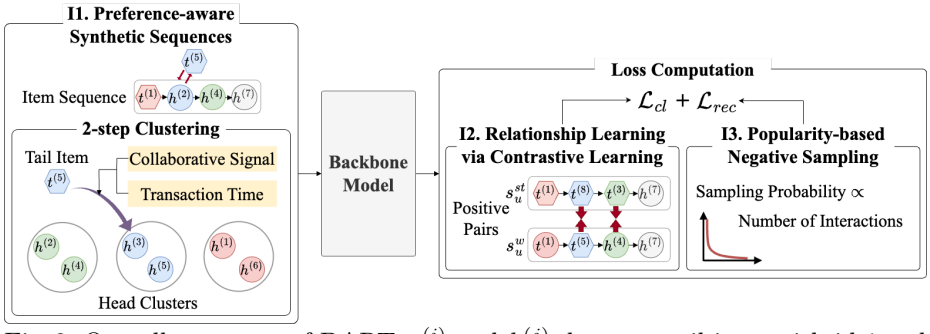

Fig. 2: Overall structure of DART. $t^{(i)}$ and $h^{(j)}$ denote a tail item with id i and head item with id j, respectively.

3 Proposed Method

3.1 Overview

There are several challenges associated with designing a sequential recommendation model that maximizes both tail accuracy and tail coverage.

C1. **Highly skewed interactions.** Models trained on skewed datasets have limited opportunities to train embeddings of tail items. How can we increase the training opportunities for tail items?

C2. **Unknown characteristics of tail items.** The limited number of interactions with tail items provides insufficient informative signals to adequately represent them. How can we accurately capture their characteristics?

C3. **Distorted information learning for tail item embeddings.** Random negative sampling distorts the embedding learning for tail items. How can we mitigate this distortion?

The main ideas of DART are summarized as follows:

I1. **Preference-aware synthetic sequence.** DART generates synthetic sequences that elevate the proportion of tail items while maintaining the integrity of the original interest.

I2. **Relationship learning via contrastive learning.** We identify similarities between tail and head items and design a contrastive learning scheme to reflect these similarities in the embedding vectors.

I3. **Popularity-based negative sampling.** We reduce the inclusion of uncertain information in the embeddings of tail items through popularity-based negative sampling.

Fig. 2 shows an overview of DART. We identify similarity relationships of items by clustering items exploiting collaborative signals and transaction time. With the cluster information, we generate synthetic sequences that mitigate the imbalance between head and tail items. We integrate these sequences with contrastive learning, which brings item embeddings in the same cluster closer. Algorithm 1 summarizes the training process of DART. After clustering items (line 1), DART iterates over sequence batches until accuracy converges (lines 2 - 3). DART synthesizes sequences by replacing α percent of items in each sequence and computes the binary cross-entropy loss $\mathcal{L}_{rec}$ (lines 4 - 5). DART performs strong

Algorithm 1 Training of DART

Input: Untrained model parameters θ, item set $\mathcal{I}$, sequence set $\mathcal{S}$, balancing hyper-parameter λ, and replacement ratios α, α_{st}, and α_w, where $0 < \alpha_w < \alpha < \alpha_{st} < 1$
Output: Trained model parameters θ
1: $\mathbb{C} \leftarrow$ Clusters of items $\mathcal{I}$ based on $\mathcal{S}$ (see Section 3.2)
2: **while** Accuracy has not converged **do**
3: **for** each sequence batch $\mathcal{S}_\mathcal{B}$ in the sequence dataset $\mathcal{S}$ **do**
4: $\mathcal{S}_\mathcal{B}^{replace} \leftarrow replace(\mathcal{S}_\mathcal{B}, \alpha)$ (see Section 3.2)
5: Compute binary cross-entropy loss $\mathcal{L}_{rec}$ with $\mathcal{S}_\mathcal{B}^{replace}$ (see Section 3.5)
6: $\mathcal{S}_\mathcal{B}^{st} \leftarrow replace(\mathcal{S}_\mathcal{B}, \alpha_{st})$ (see Section 3.3)
7: $\mathcal{S}_\mathcal{B}^{w} \leftarrow replace(\mathcal{S}_\mathcal{B}, \alpha_w)$ (see Section 3.3)
8: Compute contrastive learning loss $\mathcal{L}_{cl}$ with $\mathcal{S}_\mathcal{B}^{st}$ and $\mathcal{S}_\mathcal{B}^{w}$ (see Section 3.3)
9: Update model parameters θ by minimizing $\mathcal{L}_{rec} + \lambda\mathcal{L}_{cl}$
10: **end for**
11: **end while**

replacement with α_{st} and weak replacement with α_w on the sequence batch and computes the contrastive learning loss $\mathcal{L}_{cl}$ (lines 6 - 8). Then, DART updates the model parameters θ by minimizing the final loss $\mathcal{L}_{rec} + \lambda\mathcal{L}_{cl}$ (line 9).

3.2 Preference-aware Synthetic Sequence

How can we mitigate the imbalance ratio of head items to tail items in the dataset? To answer this question, we generate synthetic sequences by replacing head items with tail items to elevate the proportion of tail items. We preserve the preferences of the original sequences by substituting items with similar ones. Specifically, we sample α percent of head items $\eta_{\mathbf{s}_u}$ from a user's sequence and replace each with a similar tail item. We determine item similarities by clustering the items, as explained later. Given a sequence $\mathbf{s}_u = (i_1, i_2, \ldots, i_t, \ldots, i_{|\mathbf{s}_u|})$ of user u, the replacement method $replace(\cdot)$ is formulated as follows:

$$
\begin{aligned}
&replace(\mathbf{s}_u; \alpha) = (\tilde{i}_1, \tilde{i}_2, ..., \tilde{i}_t, ..., \tilde{i}_{|\mathbf{s}_u|}), \\
&\tilde{i}_t = \begin{cases} \tau & (i_t \in \eta_{\mathbf{s}_u},\ \tau \sim U(\mathcal{I}^\tau \cap \mathcal{C}(i_t))) \\ i_t & (i_t \notin \eta_{\mathbf{s}_u}) \end{cases},
\end{aligned} \tag{1}
$$

where $U(\cdot)$ denotes uniform sampling, $\mathcal{C}(i_t)$ denotes the item cluster to which item i_t belongs, and τ is a randomly sampled tail item from the cluster $\mathcal{C}(i_t)$.

Two-stage clustering. To replace head items with tail items within the same cluster, each cluster must be appropriately balanced between head and tail items. Without a sufficient distribution of both types, clusters containing only head items will not have the necessary tail items for replacement. However, a naive clustering method based on embedding distance results in the separate clustering of head and tail items (see Fig. 6 in Section 4.4). Instead, we propose two-stage clustering that balances the number of head and tail items in each cluster. First, we apply k-means clustering to the embeddings of head items. Second, tail items are assigned to the pre-formed clusters based on their distance to the centroids.

Measuring similarity. Items must be grouped based on their actual relevance to preserve the original sequences' preferences. We combine two types of dis-

tances between items for clustering: collaborative signal and time distance. The sum of the two distances after z-score normalization is used for item clustering.

Collaborative signal. We measure item similarity by the L2 distance between pre-trained item embedding vectors. To efficiently capture user-item relationships computationally, we employ a Multi-Layer Perceptron (MLP) model:

$$\hat{y}_{ui} = a(\mathbf{W}_L(\dots a(\mathbf{W}_1 \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix} + \mathbf{b}_1)\dots) + \mathbf{b}_L), \tag{2}$$

where $\hat{y}_{u,i}$ is the interaction score between user u and item i. p_u and q_i are the learnable embedding vectors for a user u and item i, respectively. $\mathbf{W}_x \in \mathbb{R}^{d_{out} \times d_{in}}$ and $\mathbf{b}_x \in \mathbb{R}^{d_{out} \times 1}$ are the weight and bias in the x^{th} linear layer, respectively. d_{in} and d_{out} denote the input and output dimensions of each linear layer, respectively. $a(\cdot)$ is the ReLU activation function, and L is the number of linear layers. After training the model, we calculate the collaborative distance $g(i, i')$ between items i and i' as follows: $g(i, i') = \|\mathbf{q}_i - \mathbf{q}_{i'}\|_2$.

Time distance. We also measure item similarity based on the interval between purchase times. Two items are related if the difference in their purchase times in a user's sequence is in a window size of w. Sliding the window across sequences, we compute the time distance $\mathbf{T}[i][i']$ between items i and i' as follows:

$$\mathbf{T}[i][i'] = \begin{cases} \frac{1}{c} & (\text{if } c \neq 0) \\ 1 & (\text{if } c = 0) \end{cases}, \tag{3}$$

where c represents the number of times that items i and i' are purchased within window w across all sequences.

3.3 Relationship Learning via Contrastive Learning

How can we accurately capture the characteristics of tail items? The scarcity of interactions with tail items results in an insufficient number of informative signals to accurately represent them. We supplement their inadequate information with the relationships of items identified by clustering. We design a positive pair for contrastive learning that brings the embeddings of items in the same cluster closer. Specifically, we generate two synthetic sequences for each user to train them as positive pairs by performing head-to-tail replacement at two levels: strong replacement with a high replacement ratio α_{st} and weak replacement with a relatively small ratio α_w. Contrastively learning these positive pairs brings item embeddings in the same cluster closer together (see Fig. 5 in Section 4.3). This results from sequences being nearly identical, with only a few differing items, leading the model to make the replaced items similar to each other. Additionally, head items with sufficient information supplement tail items with limited information when tail items become closer to head items.

Fig. 3 illustrates an example, where head items $h^{(2)}$ and $h^{(4)}$ are replaced with tail items $t^{(11)}$ and $t^{(10)}$, respectively, in the sequence $\mathbf{s}_u^{st}$, where strong replacement is applied, while only the head item $h^{(2)}$ is replaced with the tail item $t^{(10)}$ in the sequence $\mathbf{s}_u^{w}$ with weak replacement. Contrastively learning the two synthetic sequences $\mathbf{s}_u^{st}$ and $\mathbf{s}_u^{w}$ as a positive pair brings the embeddings of

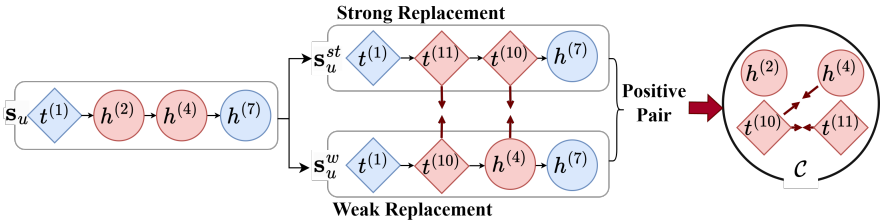

Fig. 3: DART contrastively learns the two synthetic sequences, $\mathbf{s}_u^{st}$ and $\mathbf{s}_u^{w}$, as a positive pair, bringing the embeddings of items in the same cluster closer.

items $t^{(10)}$ and $h^{(4)}$ closer, as well as those of $t^{(11)}$ and $t^{(10)}$ closer. We exploit InfoNCE [21], a general loss function in contrastive learning, as follows:

$$\mathcal{L}_{cl}(\mathbf{s}_u^{st}, \mathbf{s}_u^{w}) = -\log \frac{\exp(\phi(\mathbf{e}_{\mathbf{s}_u^{st}}, \mathbf{e}_{\mathbf{s}_u^{w}}))}{\exp(\phi(\mathbf{e}_{\mathbf{s}_u^{st}}, \mathbf{e}_{\mathbf{s}_u^{w}})) + \sum_{\substack{u' \in \mathcal{U}_\mathcal{B} \\ u' \neq u}} \sum_{s^- \in \mathcal{S}^-} \exp(\phi\left(\mathbf{e}_{\mathbf{s}_u^{st}}, \mathbf{e}_{\mathbf{s}^-}\right))}, \quad (4)$$

where $\mathbf{e}_{\mathbf{s}_u}$ is the representation vector of the sequence $\mathbf{s}_u$ from Section 2.2, $\phi(\cdot, \cdot)$ is the cosine similarity function, $\mathcal{S}^- = \{\mathbf{s}_{u'}^{st} \cup \mathbf{s}_{u'}^{w}\}$, and $\mathcal{U}_\mathcal{B}$ is the set of users in the batch $\mathcal{B}$.

3.4 Popularity-aware Negative Sampling

We address the sensitivity of tail items to distortion caused by negative sampling to prevent the model from learning inaccurate information. Even a single weight update from inaccurate negative examples significantly affects their representations because tail items have limited training data. However, many existing models sample negative items uniformly while treating all items equally. We sample negative items proportionally to their popularity to avoid the selection of tail items as negatives while encouraging the selection of popular items as negative samples. For example, it can be reasonably inferred that it does not match their preferences if a user does not purchase a popular product. In contrast, it is unreasonable to draw such a conclusion for less popular items based solely on the absence of purchase. The probability $P_{pop}(i)$ of item i being selected for negative sampling is defined as $P_{pop}(i) = \frac{\text{count}(i)}{\sum_{i \in \mathcal{I}} \text{count}(i)}$, where $count(i)$ is the total number of times that item i has been purchased in the entire dataset.

3.5 Objective Function

To learn the user-item relationship, we exploit the binary cross-entropy loss:

$$\mathcal{L}_{rec} = -\sum_{u \in \mathcal{U}_\mathcal{B}} \left\{ \log\left(\sigma(r_{u,i^+},)\right) + \sum_{i^- \in \mathcal{I}^-} \log\left(1 - \sigma(r_{u,i^-})\right) \right\}, \quad (5)$$

where σ is the sigmoid function, i^+ is the target item that user u consumed at timestamp $(m+1)$, r_{u,i^+} is the relatedness score between user u and item i^+, and $\mathcal{U}_\mathcal{B}$ is the set of users in a batch $\mathcal{B}$. The negative item i^- that user u did not consume at timestamp $(m+1)$ is sampled by the negative sampling approach described in Section 3.4. We define the final objective function for a batch $\mathcal{B}$ as $\mathcal{L}_{final} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl}$, where λ is a balancing hyperparameter that adjusts the weight of the contrastive learning loss $\mathcal{L}_{cl}$ in Equation (4).

Table 1: Summary of datasets.

Dataset	Users	Items	Interactions
MovieLens[1]	6,040	3,616	661,028
Amazon Books[2]	3,740	14,187	275,586
Yelp [3]	50,423	36,591	1,338,087

[1] https://grouplens.org/datasets/movielens/,
[2] https://nijianmo.github.io/amazon/index.html,
[3] https://www.yelp.com/dataset

4 Experiments

We performed experiments to answer the following questions:

Q1. **Performance (Section 4.2).** Does DART show high tail accuracy and tail coverage without compromising the overall accuracy?
Q2. **Ablation study (Section 4.3).** How do the main ideas of DART affect the performance?
Q3. **Comparison of clustering (Section 4.4).** How does the clustering method of DART affect the performance compared to other clustering methods?

4.1 Experimental Setup

Datasets. We use three real-world rating datasets, as summarized in Table 1. Books is from Amazon, a large e-commerce platform. Yelp is a review dataset for restaurants, and MovieLens-1M contains user ratings for movies. We filter out users and items that have fewer than five interactions.

Evaluation metrics. Following prior studies [3, 16], we employ the leave-one-out protocol, where the last item in each user's sequence is removed for testing. Accuracy is evaluated using nDCG@K [4, 10] and Tail nDCG@K; the latter measures the performance when the ground-truth item is a tail item. Diversity is measured using Coverage@K, which represents the proportion of items recommended to all users in the top-K reecommendation list relative to the entire item set, and Tail Coverage@K, which represents the proportion of recommended tail items relative to the tail item set. We set K to 10.

Baselines. We compare DART with the following baselines, where the first three focus on accuracy, while the last three focus on tail performance:

- SASRec [9] is a sequential recommendation model based on the Transformer.
- FMLP [27] is an MLP-based sequential model with learnable filters that reduce the noise information.
- BERT4Rec [23] is a sequential recommendation model that utilizes the bidirectional encoder representations from the BERT architecture.
- CoLTRec [18] is a sequential recommendation model that enhances tail accuracy through the aggregation of embeddings.
- MELT [13] is a model-agnostic sequential recommendation model that alleviates both the long-tail user and item problems.
- DCRec [26] is a state-of-the-art sequential recommendation model that mitigates popularity bias with contrastive learning.

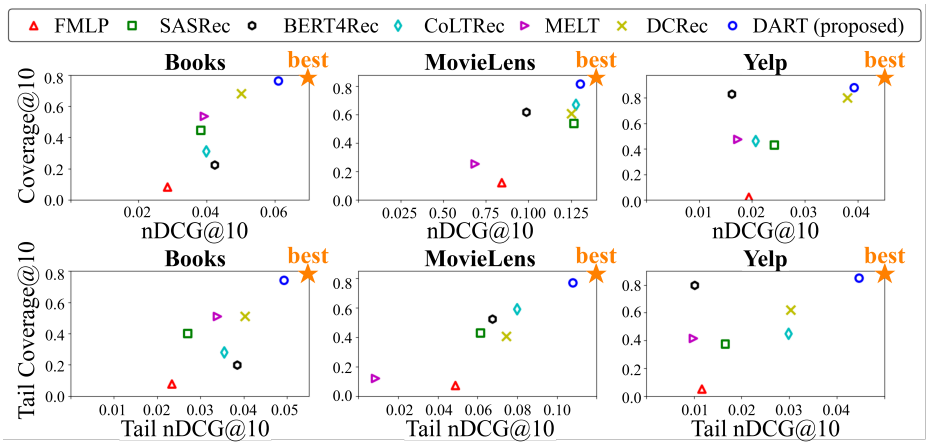

Fig. 4: Performance comparison. DART shows the best performance in all datasets, closest to the best point marked by an orange star.

Implementation. For a fair comparison, we replace CoLTRec's base model from ComiRec to SASRec, as our setting excludes category information used by ComiRec. For MELT, we use SASRec as the base model. We use public hyperparameters for existing datasets, while new datasets utilize a random search within ranges proposed by each model. The models are trained for 500 epochs, with early stopping after 30 epochs without improvement. We run five experiments with different random seeds and report the average results.

4.2 Performance (Q1)

We present a comparison between the DART and baseline models in Fig. 4. DART consistently outperforms baselines in both overall and tail performance. DART surpasses the best competitor, DCRec, with 22% higher nDCG@10 while improving Coverage@10 by 11.9% on the Books dataset. DART achieves a 35.1% improvement in tail nDCG@10 and a 30.4% increase in Tail Coverage@10 compared to the best competitor in the MovieLens dataset. These results highlight that DART not only improves the overall accuracy and tail accuracy but also enhances the aggregate-level diversity.

4.3 Ablation Study (Q2)

We evaluate the effectiveness of each component in DART by integrating them individually into the base model $SASRec$. Table 2 shows that all components improve performance across datasets. $SASRec + S$ incorporates the sequence synthesis (Section 3.2), $SASRec + C$ includes the contrastive learning (Section 3.3), and $SASRec + N$ adds the popularity-based negative sampling (Section 3.4) to $SASRec$. $SASRec + S$, which addresses the low frequency of tail items in the dataset, enhances both the overall coverage and tail coverage. $SASRec + C$ boosts tail accuracy and coverage by enhancing the quality of tail item representation through their relationships with head items. $SASRec + N$ improves tail accuracy by facilitating accurate learning of tail item embeddings.

Table 2: Ablation study of DART. All the components of DART help improve the performance. Asterisk (*) denotes higher performance than SASRec.

Dataset	Metric	Model			
		$SASRec$	$SASRec + S$	$SASRec + C$	$SASRec + N$
Books	nDCG@10	0.0383	0.0562*	0.0589*	0.0571*
	Tail nDCG@10	0.0271	0.0419*	0.0477*	0.0460*
	Head nDCG@10	0.0729	0.0916*	0.0865*	0.0845*
	Coverage@10	0.4444	0.7041*	0.7549*	0.7736*
	Tail Coverage@10	0.4010	0.6634*	0.7369*	0.7616*
MovieLens	nDCG@10	0.1267	0.1405*	0.1133	0.1266
	Tail nDCG@10	0.0271	0.0419*	0.0477*	0.0460*
	Head nDCG@10	0.1634	0.1673*	0.1106	0.1370
	Coverage@10	0.5397	0.6869*	0.9065*	0.8506*
	Tail Coverage@10	0.4274	0.6087*	0.8835*	0.8132*

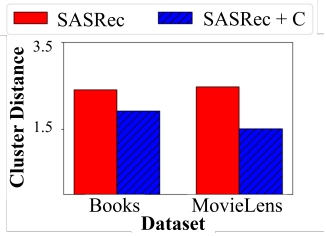

Fig. 5: The distance between item embeddings in clusters.

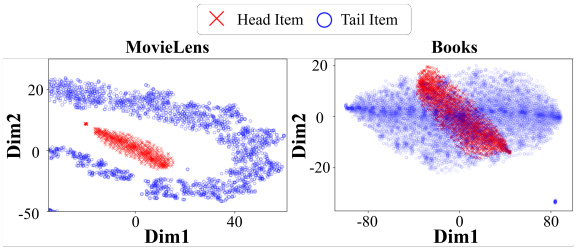

Fig. 6: t-SNE visualizations of the embeddings of head and tail items from the pretrained MLP.

Furthermore, we evaluate the effect of contrastive learning in bringing the item embeddings within the same cluster closer together. The similarity is quantified by calculating the average L2 distance between pairs of items in each cluster and averaging these distances across all clusters. Fig. 5 shows a reduction in the average distance between items in clusters after the application of the proposed contrastive learning, confirming the effectiveness of the proposed approach.

4.4 Comparison of Clustering (Q3)

We compare the proposed clustering against three baselines: naive k-means, two-stage, and random clustering. Each method replaces the clustering step in DART. k-means method performs k-means clustering on all items using pre-trained embeddings. The two-stage method performs two-stage clustering only with the pre-trained embeddings, excluding temporal distance. Random clustering divides items into k random groups. Table 3 shows that the proposed method outperforms all baselines across all metrics. This is because DART balances head and tail items in clusters by leveraging head items to enhance tail performance. Fig. 6 shows the t-SNE visualization of the item embeddings from the pre-trained MLP, showing that head and tail items tend to cluster separately. This causes k-means to group tail items together, degrading performance. Random clustering partially mitigates this issue. The two-stage approach underperforms other clustering methods owing to insufficient information in tail item embeddings, while DART solves this issue by incorporating temporal distance.

Table 3: Performance of different clustering methods. DART demonstrates the best performance. $k-m$, $2-st$, and rd denote k-means clustering, two-stage clustering, and random clustering, respectively.

Dataset	Metric	Model			
		$DART_{k-m}$	$DART_{2-st}$	$DART_{rd}$	$DART$
Books	nDCG@10	0.0552	0.0549	0.0589	**0.0611**
	Tail nDCG@10	0.0446	0.0443	0.0481	**0.0493**
	Head nDCG@10	0.0814	0.0813	0.0859	**0.0906**
MovieLens	nDCG@10	0.1226	0.1250	0.1274	**0.1305**
	Tail nDCG@10	0.1036	0.1040	0.1073	**0.1081**
	Head nDCG@10	0.1332	0.1367	0.1385	**0.1429**

5 Conclusion

We propose DART (Diversified and AccuRate Long-Tail Recommendation), a recommendation model that accurately recommends tail items while providing a broader range of items to users. DART increases the proportion of tail items by replacing head items with tail items in sequences, maintaining the original preferences. DART also strengthens the representation of tail items by proposing positive pairs considering the relationships between head and tail items for contrastive learning. We apply popularity-based negative sampling to make only trustworthy information contribute to the embedding of tail items. We show that DART achieves up to 44.7% higher Tail Coverage@10 and 47.5% higher Tail nDCG@10 compared to the best competitor.

Acknowledgments. This work was supported by Jung-Hun Foundation, Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) [No.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], and [NO.RS-2021-II212068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)]. The Institute of Engineering Research at Seoul National University provided research facilities, and the ICT at Seoul National University provides research facilities for this study. U Kang is the corresponding author.

References

1. Bai, B., Fan, Y., Tan, W., Zhang, J.: DLTSR: A deep learning framework for recommendations of long-tail web services. IEEE Trans. Serv. Comput. (2020)
2. Chen, Y., andJia Li, Z.L., McAuley, J.J., Xiong, C.: Intent contrastive learning for sequential recommendation. In: WWW (2022)
3. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: WWW (2017)
4. Jeon, H., Jang, J.G., Kim, T., Kang, U.: Accurate bundle matching and generation via multitask learning with partially shared parameters. PLOS ONE (2023)
5. Jeon, H., Kim, J., Lee, J., Lee, J., Kang, U.: Aggregately diversified bundle recommendation via popularity debiasing and configuration-aware reranking. In: PAKDD (2023)

6. Jeon, H., Kim, J., Yoon, H., Lee, J., Kang, U.: Accurate action recommendation for smart home via two-level encoders and commonsense knowledge. In: CIKM (2022)
7. Jeon, H., Lee, J., Yun, J., Kang, U.: Cold-start bundle recommendation via popularity-based coalescence and curriculum heating. In: WWW (2024)
8. Johnson, J., Ng, Y.: Enhancing long tail item recommendations using tripartite graphs and markov process. In: WI (2017)
9. Kang, W., McAuley, J.J.: Self-attentive sequential recommendation. In: ICDM (2018)
10. Kim, D., Tanwar, S., Kang, U.: Accurate multi-behavior sequence-aware recommendation via graph convolution networks. PLOS ONE (2025)
11. Kim, J., Jeon, H., Lee, J., Kang, U.: Diversely regularized matrix factorization for accurate and aggregately diversified recommendation. In: PAKDD (2023)
12. Kim, J., Kang, U.: Sequentially diversified and accurate recommendations in chronological order for a series of users. In: WSDM (2025)
13. Kim, K., Hyun, D., Yun, S., Park, C.: MELT: mutual enhancement of long-tailed user and item for sequential recommendation. In: SIGIR (2023)
14. Koo, B., Jeon, H., Kang, U.: Accurate news recommendation coalescing personal and global temporal preferences. In: PAKDD (2020)
15. Koo, B., Jeon, H., Kang, U.: PGT: news recommendation coalescing personal and global temporal preferences. Knowl. Inf. Syst. (2021)
16. Lee, J., Yun, J., Kang, U.: Towards true multi-interest recommendation: Enhanced scheme for balanced interest training. In: BigData (2024)
17. Li, J., Lu, K., Huang, Z., Shen, H.T.: Two birds one stone: On both cold-start and long-tail recommendation. In: ACM Multimedia (2017)
18. Liu, Y., Zhang, X., Zou, M., Feng, Z.: Co-occurrence embedding enhancement for long-tail problem in multi-interest recommendation. In: RecSys (2023)
19. Liu, Z., Mei, S., Xiong, C., Li, X., Yu, S., Liu, Z., Gu, Y., Yu, G.: Text matching improves sequential recommendation by reducing popularity biases. In: CIKM (2023)
20. Markwood, P.S.: The long tail: Why the future of business is selling less of more. Learn. Publ. (2010)
21. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. CoRR (2018)
22. Sreepada, R.S., Patra, B.K.: Mitigating long tail effect in recommendations using few shot learning technique. Expert Syst. Appl. (2020)
23. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: CIKM (2019)
24. Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., Cui, B.: Contrastive learning for sequential recommendation. In: ICDE (2022)
25. Yang, H., Choi, Y., Kim, G., Lee, J.: LOAM: improving long-tail session-based recommendation via niche walk augmentation and tail session mixup. In: SIGIR (2023)
26. Yang, Y., Huang, C., Xia, L., Huang, C., Luo, D., Lin, K.: Debiased contrastive learning for sequential recommendation. In: WWW (2023)
27. Zhou, K., Yu, H., Zhao, W.X., Wen, J.: Filter-enhanced MLP is all you need for sequential recommendation. In: WWW (2022)