ALTRec: Adversarial Learning for Autoencoder-based Tail Recommendation

Jixiong Liu, Dugang Liu, Weike Pan* and Zhong Ming*
College of Computer Science and Software Engineering, Shenzhen University
Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
Shenzhen, China
liujixiong@email.szu.edu.cn, dugang.ldg@gmail.com, {panweike,mingz}@szu.edu.cn

2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA) | 978-1-6654-7330-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSAA54385.2022.10032423

***Abstract*—Autoencoder-based methods have achieved significant performance on item recommendation. However, they may not perform well on tail items due to the ignorance of the items' popularity bias. As a response, in this paper, we focus on tail items and propose a novel adversarial learning method for tail recommendation (ALTRec). In our ALTRec, the generator (i.e., AutoRec) not only reconstructs the input well, but also minimizes the (any two-user) similarity difference between the input stage and the output stage to keep users' interaction relationships unchanged. And the discriminator maps the inputs and outputs of the generator to a same semantic space for scoring the similarity and maximizes the similarity difference as the target, and will identify some unsatisfactory predictions, especially on tail items. In order to preserve the similarity, the generator will pay more attention to the tail items compared with the previous autoencoder-based methods. An ablation study validates the effectiveness of preserving the two-user similarity, as well as the adversarial learning strategy in our ALTRec. Extensive experiments on three real-world datasets show that our ALTRec significantly boosts the performance on tail items compared with several state-of-the-art methods.**

***Index Terms*—Adversarial Learning, Tail Recommendation, Collaborative Filtering, Implicit Feedback**

I. Introduction

Item recommendation has been well recognized as an important solution to alleviate the information overload problem existing in various application scenarios such as e-commerce and news recommendation. Technically, collaborative filtering-based recommendation methods that model the users' tastes solely based on the (user, item) interactions have been widely explored and deployed. In particular, matrix factorization based methods and autoencoder-based methods are two important families in this branch, where the latter have been attracting more and more attention of both researchers and practitioners because of their simplicity and powerful representation ability. For example, AutoRec [1] converts a user's past interaction behaviors to a vector as input, and learns the user's preference by minimizing the reconstruction loss between the input vectors and the prediction ones. CDAE [2] introduces a dropout trick to enhance the robustness, and an additional node of a user's representation for better personalization. And Mult-VAE [3] assumes that the representation of a user should conform to a Gaussian distribution, in which the encoder is designed to learn the mean value and standard deviation, and the decoder generates the user's preferences on all the items based on the corresponding multinomial likelihood.

*: Co-corresponding authors

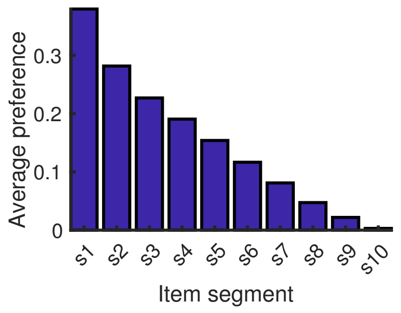


Fig. 1. Average predicted preference of AutoRec on decile items with different popularity, i.e., from the most popular segment (s1) to the least popular one (s10), of MovieLens 1M.

Although autoencoder-based item recommendation methods have achieved significant overall performance, their performance on the tail items (i.e., the unpopular ones) are usually not satisfactory [4]. The reason is that they may unconsciously emphasize the fitting on the head items (i.e., the most popular ones) while neglecting the tail ones due to the difference in the amount of training samples. We call this phenomena the popularity bias, and illustrate it in Figure 1 via a distribution of the average predicted preference over the items with different popularity in MovieLens 1M. We can see that the average preference of a well-trained AutoRec on the head items is much larger than that on the tail ones, which indicates that the model tries to minimize the loss mostly defined on the head items instead of on all the items equally.

Due to insufficient modeling of the tail items, one conjecture is that an autoencoder-based method may struggle to maintain the similarity between any two users in the input and prediction stages. But in fact, the similarity (e.g., cosine similarity) between the two stages is usually close, because the interaction history of most users is often concentrated on the head items, making it difficult for the inaccurate predicted preferences on a small number of tail items to influence the similarity a lot. We give an example in Figure 2, where values with red and black correspond to the head and tail items, respectively. Due to the dominance of the head item, user 2 and user 3 still have high similarity in both stages, despite having an inaccurate predicted preference on the tail items. However, if we pay more attention to the tail items, it can be easily recognized that the similarity between user 2 and user 3 is inconsistent in the input stage and the prediction stage. To avoid being simply revealed, the model can add confusion by adjusting the prediction mode on the tail items.

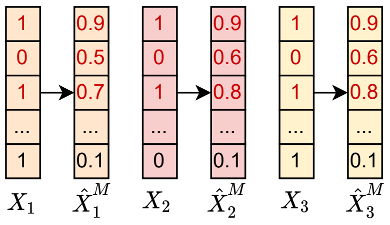


Fig. 2. An example of the input and prediction vectors on three different users, where values with red and black correspond to the head and tail items, respectively.

Based on the above description, in order to improve the impact of the predicted preferences on tail items to the similarity, we propose an adversarial learning (AL) approach for tail recommendation (TRec), i.e., ALTRec, by introducing a constraint on the similarity of two users (i.e., the two-user similarity). In particular, we map the concatenation of two users' input (or output) vectors of the generator to a same semantic space via the discriminator. In this way, the discriminator scores the similarity of the two users, which is further designed to maximize the difference between the the similarity w.r.t. the input stage and that w.r.t. the output stage of the generator, in order to expand the influence of the predicted preference on each item on the similarity. On the contrary, in order to address the issue of the popularity bias and improve the performance on the tail items, we minimize the above similarity difference via the generator aiming to treat all the items equally (instead of emphasizing on the head items only). Notice that the similarity difference will be large if the generator reconstructs the preferences well on the head items only like AutoRec. For this reason, the generator is designed to minimize the difference so as to pay more attention to the tail items. An ablation study in the experiments validates the effectiveness of the constraint and the proposed adversarial learning strategy. Meanwhile, empirical studies show the effectiveness of our ALTRec compared with several very competitive methods.

We summarize our main contributions as follows: (i) we point out the problem that most previous autoencoder-based methods usually do not perform well on tail items, which can be alleviated by keeping users' similarity in different stages unchanged; (ii) we introduce an adversarial learning strategy to further enhance the similarity constraint used for recommending tail items; and (iii) we conduct extensive comparative studies between our solution (i.e., ALTRec) and several state-of-the-art methods on three real-world datasets, and show the effectiveness of our ALTRec on tail recommendation.

II. Our Solution

A. Problem Definition

In this paper, we use $u \in \mathcal{U} = \{1, 2, ..., n\}$ to index users and $i \in \mathcal{I} = \{1, 2, ..., m\}$ to index items. For item recommendation, we convert all the (user, item) interaction entries to '1' and all the missing entries to '0', and thus obtain a binary matrix $X \in \mathbb{R}^{n \times m}$. For each user u, we use $X_u \in \mathbb{R}^m$ to represent the user u's binary vector, in which '1' means that the user has interacted with the item, and '0' otherwise. And our goal is to improve the recommendation performance on tail items.

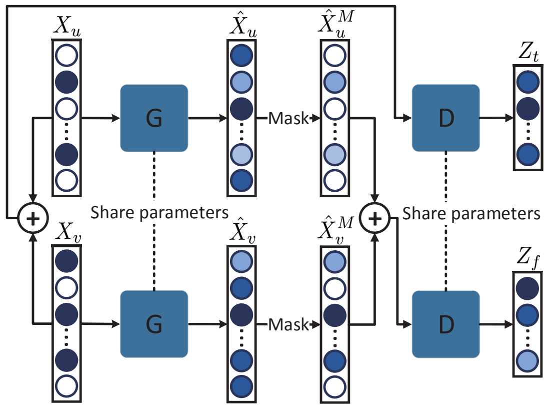


Fig. 3. Illustration of our ALTRec, where the generator (G) aims to reconstruct the input and minimize the similarity difference (of two users) between the input stage and the output stage, and the discriminator (D) is a neural network that maps the inputs from two different stages to a same semantic space and maximizes the similarity difference.

B. Overall Framework

The items' popularity bias will inevitably cause an autoencoder-based method such as AutoRec to emphasize the importance of the head items in training, resulting in insufficient training on the tail ones. Meanwhile, due to the lack of training samples w.r.t. tail items, it is difficult for the inaccurate predicted preferences on tail items to have a sufficient impact on the similarity defined on any two users. In order to address the problem, we propose an adversarial learning strategy in our ALTRec to improve the impact of a user's preference on each item on the similarity. Hence, our proposed ALTRec will pay more attention to the tail items because of insufficient modeling on them.

We illustrate our solution in Figure 3, and formulate it as follows,

$$\min_{\theta} \max_{\phi} \mathop{\mathbb{E}}_{u,v \in \mathcal{U}} f(Z_t, Z_f), \tag{1}$$

where θ and ϕ are parameters of the generator (i.e., AutoRec) and the discriminator, respectively. $Z_t = D(X_u \oplus X_v)$ or $Z_f = D(\hat{X}_u^M \oplus \hat{X}_v^M)$ is a mapped vector obtained from the discriminator, indicating the similarity between the two vectors fed to the concatenation operation $\oplus$. X_u and $\hat{X}_u^M$ are the user

u's binary vector of historical interactions and masked vector of the predicted preferences $\hat{X}_u$, i.e., $\hat{X}_u^M = \hat{X}_u \cdot X_u$. And $f(Z_t, Z_f)$ scores the similarity difference between u and v in two stages, i.e., the input stage and the output stage of the generator.

We can see that the generator G and the discriminator D play a min-max game, in which the generator aims to minimize the similarity difference in order to maintain users' interaction relationships, which helps the generator pay more attention to the tail items ignored by previous autoencoder-based methods in the optimization process. As an adversary, the discriminator maps inputs from different sources to a same semantic space to represent the similarity. Meanwhile, the discriminator aims to maximize the similarity difference, in order to maximize the impact of the predicted preference on each item in the generated vector on the similarity difference. Due to insufficient modeling on the tail items, the tail items will actually become the focused target of the adversarial optimization.

C. The Generator

As the basis of autoencoder-based methods, AutoRec takes the binary vector of a user's historical interactions as input, learns the user's representation through the encoder, and then predicts the user's preferences on all the items through the decoder. During training, it seems that the importance of the training samples on head items and tail items is the same, but in fact AutoRec always pays more attention to the most samples from the head items rather than from the tail items, which makes the performance on tail items poor.

In order to address this issue, we consider that the similarity between any two users should be maintained at different stages of the generator, so as to encourage the model to treat the training samples on the head and tail items with equal importance. Meanwhile, to further expand the influence of the predicted preference on each item on the similarity, we introduce an adversarial learning strategy so that the model will pay more attention to modeling the tail items well. Specifically, we formulate our method as follows,

$$\begin{aligned}\theta^* &= \arg\min_{\theta} - \mathop{\mathbb{E}}_{u\in\mathcal{U}} [X_u \log(\hat{X}_u) + (1 - X_u)\log(1 - \hat{X}_u)] \\ &+ \lambda_g \mathcal{R}(\theta) + \alpha \mathbb{E}_{u,v\in\mathcal{U}} f(Z_t, Z_f), \qquad (2)\end{aligned}$$

where the first term and second term are the reconstruction loss and regularization term, respectively, inheriting from AutoRec. And we introduce a hyper-parameter λ_g to control the importance of the regularization. Meanwhile, another hyper-parameter α is used to control the importance of the adversarial loss, i.e., the last term, which measures the similarity difference between two users. In order to reduce the complexity to meet the challenge of the large number of users, we select the most active users denoted as $\mathcal{U}_a$, and then repeat this to obtain $|\mathcal{U}|$ active users $\mathcal{U}_b$. After that, we sample a user u from $\mathcal{U}_b$ and another user v from $\mathcal{U} = \{1, 2, ..., n\}$, and reformulate the last term as follows,

$$\begin{aligned}&\mathbb{E}_{u,v\in\mathcal{U}} f(Z_t, Z_f) \\ \approx\ & \frac{1}{n} \sum_{u\in\mathcal{U}_b, v\in\mathcal{U}} f(D(X_u \oplus X_v), D(\hat{X}_u^M \oplus \hat{X}_v^M)) \\ =\ & \frac{1}{n} \sum_{u\in\mathcal{U}_b, v\in\mathcal{U}} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2, \quad (3)\end{aligned}$$

where we take the concatenation vector $X_u \oplus X_v$ (or $\hat{X}_u^M \oplus \hat{X}_v^M$) as the discriminator's input, and then obtain a mapped vector Z_t (or Z_f), representing the similarity of user u and user v. Meanwhile, we only consider the predicted preferences on ground truth items, i.e., $\hat{X}_u^M = \hat{X}_u \cdot X_u$ and $\hat{X}_v^M = \hat{X}_v \cdot X_v$, in order to eliminate the impact from the others. And we use a variant of Euclidean distance to measure the similarity in our experiment. Obviously, sampling u from fixed $\mathcal{U}_b$ and v from $\mathcal{U}$ will pay more attention to the training samples on $\mathcal{U}_b$, which introduces user bias leading to unbalanced training. To avoid it, we use a stop gradient strategy[1] that stops gradient from adversarial loss on $u \in \mathcal{U}_b$ from being back-propagated to the generator so as to treat all the users fairly.

D. The Discriminator

Different from other adversarial learning based recommendation algorithms [5]–[11] whose discriminator classifies the inputs of different sources into different labels, the discriminator in our ALTRec maps the inputs from different sources to a same semantic space so as to score the similarity [12]. Notice that the generator's collaborator in [12] maps the inputs to a same semantic space but the discriminator still assigns different labels. However, our discriminator takes the vectors of two users in the input stage or the output stage of the generator as input, and uses the learned mapping vector to represent the similarity of the two users at the corresponding stage. And the goal of the discriminator is to maximize the similarity difference between these two stages, in order to help the generator identify which item is not sufficiently modeled. To achieve this, we formulate the objective of the discriminator as follows,

$$\begin{aligned}&\phi^* \\ =\ & \arg\min_{\phi} - \mathop{\mathbb{E}}_{u,v\in\mathcal{U}} f(Z_t, Z_f) \\ & \underbrace{+\lambda_d \mathop{\mathbb{E}}_{\tilde{X}\in\mathbb{P}_{\tilde{X}}} [(\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2]}_{\text{Gradient penalty}} \\ \approx\ & \arg\min_{\phi} - \frac{1}{n} \sum_{u\in\mathcal{U}_b, v\in\mathcal{U}} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2 \\ & + \lambda_d \mathbb{E}_{\tilde{X}\in\mathbb{P}_{\tilde{X}}} [(\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2], \qquad (4)\end{aligned}$$

where the first term is to maximize the similarity difference between the two stages. And the second term is to enforce the Lipschitz constraint borrowed from WGAN-GP [13], which

[1]https://www.tensorflow.org/api_docs/python/tf/stop_gradient

Algorithm 1 The algorithm of ALTRec.

```
Input: Implicit feedback matrix $X \in \{0,1\}^{n \times m}$.
Initialize: Parameters of the generator and discriminator.
1:  repeat
2:      for d-steps do
3:          All users set $\mathcal{U}$ in reverse active order.
4:          Select the most active users $\mathcal{U}_a$ and shuffle them.
5:          Repeat $\mathcal{U}_a$ to obtain $\mathcal{U}_b$ until $|\mathcal{U}_b| = |\mathcal{U}|$
6:          for $u \in \mathcal{U}_b, v \in \mathcal{U}$ do
7:              Obtain true preference vectors $X_u$ and $X_v$.
8:              Obtain predicted preference $\hat{X}_u^M$ and $\hat{X}_v^M$.
9:              Sample $\epsilon \in U(0,1)$.
10:             Obtain $\tilde{X} = \epsilon(X_u \oplus X_v) + (1-\epsilon)(\hat{X}_u^M \oplus \hat{X}_v^M)$.
11:             Optimize $\phi$ by minimizing Eq.(4).
12:         end for
13:     end for
14:     for g-steps do
15:         All users set $\mathcal{U}$ in reverse active order.
16:         Shuffle $\mathcal{U}_a$.
17:         Repeat $\mathcal{U}_a$ to obtain $\mathcal{U}_b$ until $|\mathcal{U}_b| = |\mathcal{U}|$
18:         for $u \in \mathcal{U}_b, v \in \mathcal{U}$ do
19:             Obtain true preference vectors $X_u$ and $X_v$.
20:             Obtain $\hat{X}_u$, $\hat{X}_v$, $\hat{X}_u^M$ and $\hat{X}_v^M$.
21:             Optimize $\theta$ by minimizing Eq.(2).
22:         end for
23:     end for
24: until convergence
```

adds the constraint with a penalty on the gradient norm to generate more stable gradients. Specifically, we first randomly generate a number ϵ from a uniform distribution $U(0,1)$, and then add a penalty on the gradient of the mapping vector $D(\tilde{X})$ w.r.t. $\tilde{X} = \epsilon(X_u \oplus X_v) + (1-\epsilon)(\hat{X}_u^M \oplus \hat{X}_v^M)$. In the experiments, we use a neural network with only one hidden layer with 200 nodes as our discriminator, and use the sigmoid activation function. We take the output of the hidden layer as the mapping vector, i.e., $D(X_u \oplus X_v) = \sigma(W(X_u \oplus X_v) + b)$.

E. Discussions

We depict the whole algorithm in Algorithm 1. Similar to other adversarial learning algorithms, we alternatively update the parameters of the discriminator and the generator using the Adam optimizer. As far as we know, we are the first to adopt the adversarial learning strategy to maintain the similarity between two users at two different stages of the generator in order to improve the autoencoder-based recommendation performance on tail items.

In the discriminator step, we randomly pick up a user u from $\mathcal{U}_b$ and a user v from $\mathcal{U}$ to construct a pair (u, v), and then obtain two mapping vectors $D(X_u \oplus X_v)$ and $D(\hat{X}_u^M \oplus \hat{X}_v^M)$. Finally, we optimize the discriminator by maximizing the similarity difference. In this way, the discriminator will try to identify the items that are not well modeled (resulting in a large similarity difference), which thus guides the generator to pay more attention to them in the next round. In fact, the generator (i.e., AutoRec) always pays more attention to the learning on head items rather than on the tail ones. Therefore, the discriminator is mainly to guide the generator to pay more attention to the learning on the tail items in the next round. In the generator step, the generator needs to truly treat all the items equally instead of emphasizing more on the head items, in order to minimize the similarity difference. To reach it, the generator will pay more attention to the tail items that are not well modeled compared with the head items. In this way, the generator can better capture a user's preferences on the tail items.

III. Experiments

In this section, we conduct experiments on three real-world datasets, aiming to study the following three research questions.

- RQ1: Does our ALTRec outperform the state-of-the-art recommendation methods for recommendation of tail items?
- RQ2: How does our ALTRec perform on users with different levels of activity?
- RQ3: What is the impact of the key hyper-parameters on our ALTRec?

A. Datasets

We conduct empirical studies on three public datasets, including MovieLens[2] 100K (ML100K), MovieLens 1M (ML1M) and Anime[3]. Specifically, we convert all the ratings that are larger than or equal to '1' to '1' and the others to '0' so as to obtain a binary matrix for item recommendation [8], [14]. We put the statistics of each dataset in Table I. Meanwhile, for each dataset, we randomly split it into three parts, i.e., 60% for training, 20% for validation and the remaining 20% for test. We take the top 10% and 20% most popular items as head items and the rest as the tail ones [9], [15]. In the validation data, we further obtain a tail data by removing the top 10% most popular items, which is used for parameter tuning. Similarly, in the test data, we obtain two tail data by removing the top 10% and 20% most popular items, respectively.

TABLE I
STATISTICS OF THE DATASETS USED IN THE EXPERIMENTS.

Dataset	User #	Item #	Interaction #	Sparsity
MovieLens 100K	943	1,605	99,894	93.40%
MovieLens 1M	6,040	3,648	1,000,117	95.46%
Anime	69,600	9,475	6,336,619	99.04%

[2]https://grouplens.org/datasets/movielens/
[3]https://www.kaggle.com/CooperUnion/anime-recommendations-database

B. Evaluation Metrics

We use four ranking-oriented evaluation metrics, including precision (P@K), recall (R@K), normalized discounted cumulative gain (N@K), and mean reciprocal rank (M@K), where $K \in \{5, 20\}$.

C. Baselines

In order to study the effectiveness of our ALTRec on tail items, we compare it with two factorization-based methods, two autoencoder-based methods, and two adversarial learning methods.

- Bayesian personalized ranking with matrix factorization (BPR-MF) [16] is a well-known factorization-based method based on pairwise preference assumption, i.e., a user prefers an interacted item to an un-interacted one.
- Factored item similarity model (FISM) [17] learns the similarity between two items via the inner product of their latent representations, which are then aggregated for predicting the preference of a user to an item.
- AutoRec [1] is an autoencoder-based method, in which the encoder takes the binary vector of a user's historical interactions as input to learn the latent representation of the user, and then the decoder predicts the user's preferences to all the items.
- Variational autoencoders for collaborative filtering (Mult-VAE) [3] assumes that a user's latent representation should obey a Gaussian distribution, and thus the encoder learns the mean value and standard deviation of the distribution, and the decoder predicts the preferences based on multinomial likelihood.
- CFGAN [8] is a vector-wise training method, in which the generator uses an autoencoder to generate the user's preference vector on all the items, while the discriminator assigns different labels to the user's binary input of his/her historical interactions and the generated vector.
- LongTailGAN [9] first divides the items into head and tail items, and then calculates the associations between any two items by the number of users who have interacted with them. The generator aims to reproduce the associations by selecting some un-interacted tail items with higher associations to the interacted tail ones as candidates, while the discriminator aims to correctly classify the selected un-interacted items and the truly interacted ones. We can see that the loss of the adversarial part only comes from the selected items, and thus the remaining un-interacted tail items will not be updated.

D. Implementation Details

For parameter configuration, we use the performance of N@5 on the tail validation data. Meanwhile, we adopt an early-stop strategy with a predefined threshold 50 for the number of iterations.

For factorization-based methods, we adopt the SGD optimizer and tune the regularization coefficient and learning rate from {1e-4, 1e-3, 1e-2, 1e-1}, and the dimension of latent feature vectors from {50, 150, 250, 350}, independently. For the other methods, we mainly follow the guidance of the original papers. Specifically, we use the Adam optimizer with a learning rate 1e-4 for LongTailGAN and 1e-3 for the other methods. And we fix the hidden layer with 350 nodes for AutoRec, Mult-VAE and the generators of the adversarial learning methods. In order to avoid overfitting, we select the regularization coefficient from {1e-4, 1e-3, 1e-2, 1e-1} for AutoRec and Mult-VAE. For the adversarial learning methods, we fix g_steps and d_steps as 10 for LongTailGAN to keep the original setting, and 5 for the other methods. For CFGAN, we use the ZP strategy and tune the sample ratio of ZR from {10%, 30%, 50%}, the sample ratio of PM from {50%, 70%, 90%}, the coefficient of zero-reconstruction loss from {1e-2, 1e-1, 1}, and the regularization coefficient for the generator and the discriminator from {1e-4, 1e-3, 1e-2, 1e-1} independently. For LongTailGAN, we select the dropout rate for the discriminator and the generator from {0.1, 0.2, 0.3, 0.4, 0.5} independently, the coefficient of the adversarial term in the generator from {1e-2, 1e-1, 1, 10, 50, 100}. For our ALTRec, we tune the coefficient of the gradient penalty λ_d from {1e-1, 1, 1e1, 1e2}, the regularization coefficient λ_g in the generator from {1e-4, 1e-3, 1e-2, 1e-1}, the coefficient of the adversarial term α in the generator from {1e-2, 1e-1, 1, 10, 50, 100}, and $|\mathcal{U}_a|$ from {100, 300, 500}.

Note that the source codes are available at https://github.com/LiuJiXiong/ALTRec.

E. Performance Comparison (RQ1)

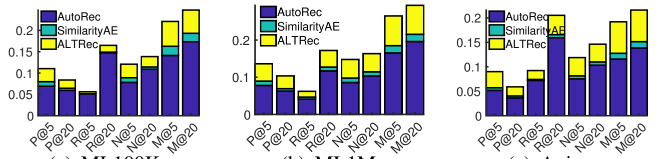
We report the main results of our ALTRec and other methods in Table II. Firstly, we can see that our ALTRec achieves the best performance compared with other methods in all cases. In particular, we choose AutoRec as the generator in our ALTRec, and find that our ALTRec performs much better than AutoRec, which indicates that the constraint of maintaining the similarity of two users through an adversarial mechanism can indeed effectively improve the recommendation performance on tail items. Secondly, Mult-VAE performs much better than most methods on ML1M and Anime except our ALTRec, which indicates that it is helpful to improve the performance on tail items by further restricting the user representation distribution to a Gaussian distribution. Because Mult-VAE performs relatively poorly on the small data, the performance of Mult-VAE does not exceed AutoRec on ML100K. Even if the sparsity of the data increases, FISM shows a relatively stable performance, while the performance of BPR-MF becomes very poor, which indicates that BPR-MF is easier to be influenced by the sparsity of the dataset. For the adversarial learning methods, LongTailGAN can not beat AutoRec on all the datasets and performs poorly on ML100K, because it mainly focuses the coverage of the tail items rather than on accuracy. Meanwhile, the adversarial loss term in LongTailGAN only focuses on a small part of the tail items, which is highly associated with the interacted tail items measured by the number of users who interact with them [9]. This will result in lack of attention on most of the remaining tail items. As for CFGAN, it mainly relies on the adversarial loss to optimize the

TABLE II
RECOMMENDATION PERFORMANCE ON TAIL ITEMS (BY REMOVING THE TOP 10% AND 20% MOST POPULAR ITEMS) IN THE TEST DATA OF MOVIELENS 100K (ML100K), MOVIELENS 1M (ML1M) AND ANIME. THE BEST RESULTS ARE MARKED IN BOLD. NOTICE THAT IMP.(%) DENOTES THE IMPROVEMENT OF OUR ALTREC COMPARED WITH THE SECOND BEST METHOD.

Dataset	Tail items	Method	P@5	P@20	R@5	R@20	N@5	N@20	M@5	M@20
ML100K	w/o top 10%	BPR-MF	0.0456	0.0478	0.0226	0.1098	0.0478	0.0759	0.0929	0.1253
		FISM	0.0584	0.0520	0.0262	0.1038	0.0633	0.0811	0.1202	0.1474
		CFGAN	0.0375	0.0365	0.0137	0.0515	0.0394	0.0478	0.0798	0.1013
		LongTailGAN	0.0106	0.0117	0.0032	0.0162	0.0109	0.0146	0.0233	0.0352
		AutoRec	0.0692	0.0594	0.0513	0.1474	0.0781	0.1089	0.1412	0.1734
		Mult-VAE	0.0649	0.0564	0.0384	0.1295	0.0692	0.0952	0.1281	0.1567
		ALTRec	**0.1104**	**0.0837**	**0.0561**	**0.1649**	**0.1208**	**0.1386**	**0.2210**	**0.2477**
		Imp.(%)	59.54%	40.91%	9.36%	11.87%	54.67%	27.27%	56.52%	42.85%
	w/o top 20%	BPR-MF	0.0160	0.0198	0.0100	0.0493	0.0175	0.0319	0.0342	0.0529
		FISM	0.0163	0.0187	0.0099	0.0508	0.0169	0.0306	0.0317	0.0475
		CFGAN	0.0077	0.0057	0.0055	0.0121	0.0090	0.0103	0.0175	0.0242
		LongTailGAN	0.0072	0.0060	0.0023	0.0101	0.0076	0.0085	0.0175	0.0231
		AutoRec	0.0261	0.0268	0.0317	0.1028	0.0322	0.0584	0.0503	0.0729
		Mult-VAE	0.0235	0.0248	0.0219	0.0858	0.0274	0.0493	0.0478	0.0685
		ALTRec	**0.0594**	**0.0469**	**0.0384**	**0.1217**	**0.0671**	**0.0854**	**0.1299**	**0.1534**
		Imp.(%)	127.59%	75.00%	21.14%	18.39%	108.39%	46.23%	158.25%	110.43%
ML1M	w/o top 10%	BPR-MF	0.0673	0.0540	0.0266	0.0818	0.0720	0.0804	0.1429	0.1685
		FISM	0.0609	0.0574	0.0247	0.0910	0.0636	0.0821	0.1192	0.1466
		CFGAN	0.0191	0.0134	0.0065	0.0149	0.0214	0.0190	0.0482	0.0575
		LongTailGAN	0.0651	0.0568	0.0282	0.0936	0.0694	0.0851	0.1332	0.1598
		AutoRec	0.0779	0.0627	0.0411	0.1170	0.0858	0.1034	0.1654	0.1960
		Mult-VAE	0.0957	0.0758	0.0452	0.1289	0.1038	0.1195	0.1931	0.2218
		ALTRec	**0.1362**	**0.1036**	**0.0621**	**0.1720**	**0.1478**	**0.1635**	**0.2649**	**0.2934**
		Imp.(%)	42.32%	36.68%	37.39%	33.44%	42.39%	36.82%	37.18%	32.28%
	w/o top 20%	BPR-MF	0.0343	0.0260	0.0188	0.0521	0.0389	0.0450	0.0794	0.0960
		FISM	0.0198	0.0220	0.0104	0.0456	0.0210	0.0333	0.0379	0.0533
		CFGAN	0.0008	0.0042	0.0001	0.0060	0.0005	0.0045	0.0009	0.0077
		LongTailGAN	0.0274	0.0256	0.0169	0.0590	0.0308	0.0432	0.0586	0.0762
		AutoRec	0.0319	0.0285	0.0253	0.0810	0.0362	0.0547	0.0664	0.0880
		Mult-VAE	0.0422	0.0364	0.0293	0.0879	0.0483	0.0657	0.0889	0.1107
		ALTRec	**0.0697**	**0.0564**	**0.0410**	**0.1252**	**0.0772**	**0.0976**	**0.1415**	**0.1678**
		Imp.(%)	65.17%	54.95%	39.93%	42.43%	59.83%	48.55%	59.17%	51.58%
Anime	w/o top 10%	BPR-MF	0.0310	0.0248	0.0273	0.0822	0.0374	0.0534	0.0670	0.0841
		FISM	0.0521	0.0351	0.0544	0.1130	0.0710	0.0853	0.1162	0.1325
		CFGAN	0.0024	0.0009	0.0019	0.0023	0.0039	0.0029	0.0090	0.0094
		LongTailGAN	0.0443	0.0318	0.0419	0.0986	0.0562	0.0706	0.0912	0.1074
		AutoRec	0.0518	0.0364	0.0721	0.1594	0.0753	0.1033	0.1158	0.1385
		Mult-VAE	0.0686	0.0423	0.0769	0.1510	0.0964	0.1134	0.1574	0.1759
		ALTRec	**0.0901**	**0.0590**	**0.0923**	**0.2052**	**0.1188**	**0.1462**	**0.1920**	**0.2160**
		Imp.(%)	31.34%	39.48%	20.03%	35.89%	23.24%	28.92%	21.98%	22.80%
	w/o top 20%	BPR-MF	0.0078	0.0078	0.0093	0.0370	0.0097	0.0190	0.0160	0.0236
		FISM	0.0177	0.0131	0.0253	0.0590	0.0261	0.0365	0.0387	0.0473
		CFGAN	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	0.0003
		LongTailGAN	0.0138	0.0115	0.0168	0.0468	0.0179	0.0275	0.0273	0.0357
		AutoRec	0.0129	0.0113	0.0251	0.0722	0.0208	0.0368	0.0281	0.0386
		Mult-VAE	0.0231	0.0162	0.0367	0.0851	0.0361	0.0514	0.0535	0.0651
		ALTRec	**0.0389**	**0.0283**	**0.0555**	**0.1420**	**0.0550**	**0.0819**	**0.0818**	**0.1000**
		Imp.(%)	68.40%	74.69%	51.23%	66.86%	52.35%	59.34%	52.90%	53.61%

model, and does not introduce a reconstruction loss like other adversarial learning methods, which increases the difficulty of optimization. In addition, like other collaborative filtering models, CFGAN will unconsciously pay more attention to the head items and ignore the tail items, resulting in poor performance on tail items as expected.

In order to verify the effectiveness of the adversarial strategy in improving the recommendation performance on tail items, we conduct an ablation study. Specifically, we implement a model called SimilarityAE, which consists of two parts, a generator and a similarity measurement component S. And



(a) ML100K (b) ML1M (c) Anime

Fig. 4. Recommendation performance of AutoRec, SimilarityAE and our ALTRec on tail items (by removing the top 10% most popular items in the test data) for ablation study. Notice that the performance of SimilarityAE is represented by the stack of blue and green, and the performance of our ALTRec is represented by the whole stack of blue, green and yellow.

the goal of S is to map the inputs from different sources to the same semantic space like the discriminator (D) in our ALTRec, but it aims to minimize the similarity difference as a collaborator of the generator. In other words, SimilarityAE has the same structure as our ALTRec, but we replace the discriminator in ALTRec with S. Notice that the goal of S is to minimize the similarity difference and update itself simultaneously with the generator. Therefore, we can verify how effective the difference between them (whether to use the adversarial strategy) is in improving the performance of the tail item recommendation. From the results in Figure 4, we can see that SimilarityAE performs a bit better than AutoRec, which shows that the constraint of maintaining the similarity between two users is helpful on the performance of the tail items. Meanwhile, our ALTRec performs better than SimilarityAE by a large margin, which shows the effectiveness of our proposed adversarial learning strategy.

F. *Users with Different Levels of Activity (RQ2)*

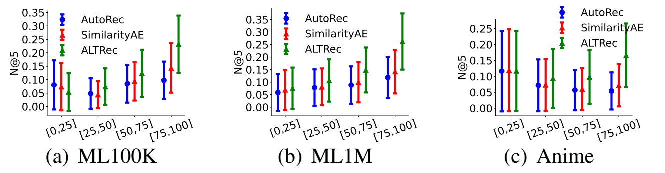


Fig. 5. Recommendation performance of AutoRec, SimilarityAE and our ALTRec on users with different activity levels on tail items (by removing the 10% most popular items in the test data). We divide users into four segments w.r.t. their activity in the training data, i.e., from the most inactive segment [0%, 25%] to the most active one [75%, 100%].

From the main results in Table II, we find that our ALTRec performs better than the state-of-the-art methods. In this subsection, we study the performance improvement of our ALTRec compared with AutoRec and SimilarityAE on users with different levels of activity. Specifically, we divide users into four segments according to their activity in the training data, including [0%, 25%], [25%, 50%], [50%, 75%] and [75%, 100%] from the least active to the most active. We report the results in Figure 5. On all the datasets, our ALTRec improves the performance more with the increase of the activity of the user segment, because users in an active segment have more training samples to express their preferences, which makes the preference prediction on the tail items more accurate. Compared with the performance improvement of the most inactive segment on ML1M, our ALTRec and SimilarityAE have no improvement on Anime, because each user interacts with at least 20 items on ML1M but only 1 item on Anime, which makes it very difficult to improve the performance by constraining the similarity between users in the segment. We can also find that the performance of SimilarityAE and our ALTRec decrease on the most inactive segment on ML100K, which may be caused by these users being more likely to interact with the head items.

G. *Impact of Key Hyper-Parameters (RQ3)*

In order to adjust the importance of the adversarial loss term in Eq.(2), we introduce a coefficient α. Meanwhile, for efficiency, we select u from the set of the most active users $\mathcal{U}_a$, and then randomly sample another user v from $\mathcal{U}$ so as to construct a (user, user) pair. In particular, in this subsection, we study the impact of these two key hyper-parameters, i.e., α and $|\mathcal{U}_a|$, on the performance, which are shown in Figure 6 and Figure 7, respectively.

From Figure 6, we can see that the performance on the tail items gradually improves and achieves the best when α equals to 50 on ML1M, and the performance achieves the best when α equals to 100 on the other two datasets. As Figure 7 shows, the model performance varies greatly with the changes of $|\mathcal{U}_a|$ on different datasets. We can see that the performance differs less with different values of $|\mathcal{U}_a|$ on ML100K and ML1M, while it is very different on Anime. The average number of items interacted by the users in Anime is smaller than that in the other two datasets, which makes it not conducive to improve the performance if more users are fixed as $\mathcal{U}_a$ but the interaction data of these added users is essentially sparse.

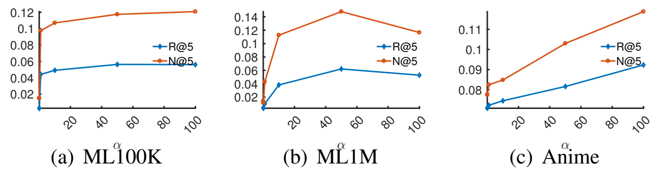


Fig. 6. Recommendation performance of our ALTRec with different values of $\alpha \in \{0.01, 0.1, 1, 10, 50, 100\}$.

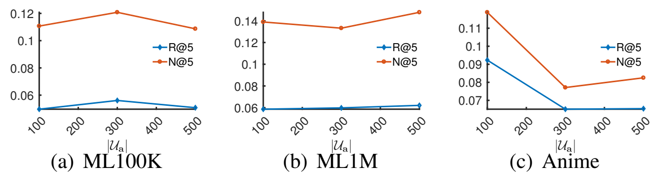


Fig. 7. Recommendation performance of our ALTRec with different values of $|\mathcal{U}_a| \in \{100, 300, 500\}$.

IV. Related Work

For item recommendation, factorization-based methods have shown very competitive performance. For example, BPR-MF [16] assumes that a user prefers an interacted item to an un-interacted one, and FISM [17] learns the similarity between any two items and represents a user by the aggregated representation of the past interacted items of the user. Besides the factorization-based methods, autoencoder-based methods have also been very popular due to its representation power. AutoRec [1] takes a binary vector for a user's historical interactions as input and infers the user's representation by the encoder, and then generates the preferences on all the items via the decoder. For improved robustness and personalization, CDAE [2] corrupts its input and introduces an embedding matrix for the users. Mult-VAE [3] introduces a distribution for the hidden representation, which is known to be more effective in reconstruction and prediction. Due to the strong generalization ability of GAN [18], [19], adversarial learning based methods have been received much attention from the community of recommender systems [20]–[24]. The discriminator in IRGAN [5] and APL [6] assigns a different reward for each generated item, while the generator selects the high-quality items from the candidate pool aiming to obtain more reward. CFGAN [8] adopts an autoencoder as the generator to generate preferences on all the items, and the discriminator assigns different labels for different sources. VAEGAN [11] introduces adversarial variational Bayes (AVB) to relax the constraint of the inferred representation in Mult-VAE, an auxiliary discriminator to reduce the reconstruction loss, and a contractive loss to enhance the robustness. To alleviate the vulnerability of the models when encountering adversarial samples [25], APR [26] and ACAE [27] aim to maintain the model's robustness by defending the challenge from the adversarial perturbations.

Although many methods perform well for item recommendation, few of them are designed to improve the performance on tail items [28], [29]. Among the methods for tail recommendation, [30] decomposes the (user, item) interaction matrix into two parts, one for head items and the other for tail ones, and then models them separately. Meanwhile, it integrates the users' side information. [15] clusters the tail items via the similarity of items' content vectors, and then the model replaces the tail items in the original sequence with cluster labels, and constructs a pseudo ground truth sequence to train the model. [9] first divides the items into head and tail ones, and then calculates the associations between any two items by the number of users who have interacted with them. And the generator aims to reproduce the associations by selecting some un-interacted tail items with higher associations to the interacted tail ones as the potential interactions. However, the above two methods mainly focus on improving the diversity of the recommendation results instead of the recommendation accuracy. There are also some algorithms that address the long-tail phenomenon by data imputation. For example, [31] is devised to augment the explicit ratings and [4] is proposed to generate virtual and plausible neighbors for users and items to address the cold-start problems.

V. Conclusions and Future work

In this paper, we focus on recommendation of tail items (or tail recommendation, TRec) and propose a novel adversarial learning (AL) method called ALTRec. In particular, we introduce a constraint to retain the two-user similarity at the input stage and the output stage of the generator. Moreover, we map the inputs and outputs of the generator to a same semantic space via the discriminator so as to represent the two-user similarity and maximize the similarity difference, which will in turn guide the generator to treat all the items equally. Importantly, the generator will then model the tail items well (instead of largely focusing on the head items only in previous autoencoder-based methods) in order to minimize the two-user similarity difference. To validate the effectiveness of the proposed constraint and the adversarial learning strategy, we compare our ALTRec with SimilarityAE and AutoRec [1], and find that SimilarityAE performs better than AutoRec indicating the usefulness of the proposed constraint, and our ALTRec further outperforms SimilarityAE by a large margin showcasing the merit of the adversarial learning strategy. Meanwhile, we find that our ALTRec performs significantly better than other state-of-the-art methods for tail item recommendation.

In the future, we are interested in studying different user pairing approaches, as well as other advanced backbone models such as Mult-VAE [3]. Moreover, we plan to study privacy protection mechanisms in designing federated tail recommendation methods with users' implicit feedback [32].

Acknowledgment

We thank the support of National Natural Science Foundation of China Nos. 62172283 and 61836005.

References

[1] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web Companion*, 2015, pp. 111–112.

[2] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, 2016, pp. 153–162.

[3] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 689–698.

[4] D. Chae, J. Kim, D. H. Chau, and S. Kim, "AR-CF: Augmenting virtual users and items in collaborative filtering for addressing cold-start problems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1251–1260.

[5] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 515–524.

[6] Z. Sun, B. Wu, Y. Wu, and Y. Ye, "APL: Adversarial pairwise learning for recommender systems," *Expert Syst. Appl.*, vol. 118, pp. 573–584, 2019.

[7] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 2508–2515.

[8] D. Chae, J. Kang, S. Kim, and J. Lee, "CFGAN: A generic collaborative filtering framework based on generative adversarial networks," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 137–146.

[9] A. Krishnan, A. Sharma, A. Sankar, and H. Sundaram, "An adversarial approach to improve long-tail performance in neural collaborative filtering," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1491–1494.

[10] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 2852–2858.

[11] X. Yu, X. Zhang, Y. Cao, and M. Xia, "VAEGAN: A collaborative filtering framework based on adversarial variational autoencoders," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4206–4212.

[12] M. Amodio and S. Krishnaswamy, "TraVeLGAN: Image-to-image translation by transformation vector learning," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8983–8992.

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2017, pp. 5767–5777.

[14] M. Volkovs and G. W. Yu, “Effective latent models for binary feedback in recommender systems,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, R. Baeza-Yates, M. Lalmas, A. Moffat, and B. A. Ribeiro-Neto, Eds., 2015, pp. 313–322.

[15] Y. Kim, K. Kim, C. Park, and H. Yu, “Sequential and diverse recommendation with long tail,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 2740–2746.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[17] S. Kabbur, X. Ning, and G. Karypis, “FISM: Factored item similarity models for top-N recommender systems,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 659–667.

[18] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[19] A. Jabbar, X. Li, and B. Omar, “A survey on generative adversarial networks: Variants, applications, and training,” *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–49, 2022.

[20] Y. Lin, Z. Xie, B. Xu, K. Xu, and H. Lin, “Info-flow enhanced GANs for recommender,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1703–1707.

[21] Y. Zhou, J. Xu, J. Wu, Z. T. Nasrabadi, E. Körpeoglu, K. Achan, and J. He, “PURE: Positive-unlabeled recommendation with generative adversarial network,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 2409–2419.

[22] C. Zhang, J. Li, J. Wu, D. Liu, J. Chang, and R. Gao, “Deep recommendation with adversarial training,” *IEEE Transactions on Emerging Topics in Computing*, 2022.

[23] C. Li, M. Zhao, H. Zhang, C. Yu, L. Cheng, G. Shu, B. Kong, and D. Niu, “RecGURU: Adversarial learning of generalized user representations for cross-domain recommendation,” in *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 2022, pp. 571–581.

[24] H. Liu, N. Zhao, X. Zhang, H. Lin, L. Yang, B. Xu, Y. Lin, and W. Fan, “Dual constraints and adversarial learning for fair recommenders,” *Knowledge-Based Systems*, p. 108058, 2022.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *CoRR*, vol. abs/1412.6572, 2014.

[26] X. He, Z. He, X. Du, and T. Chua, “Adversarial personalized ranking for recommendation,” in *Proceedings of the 41st International ACM SIGIR Conference on Research*, 2018, pp. 355–364.

[27] F. Yuan, L. Yao, and B. Benatallah, “Adversarial collaborative autoencoder for top-N recommendation,” in *Proceedings of the 2019 International Joint Conference on Neural Networks*, 2019, pp. 1–8.

[28] X. Ma, T. Qian, Y. Liang, K. Sun, H. Yun, and M. Zhang, “Enhancing graph convolution network for novel recommendation,” in *Proceedings of the 27th International Conference on Database Systems for Advanced Applications*, 2022, pp. 69–84.

[29] R. Shrivastava, D. S. Sisodia, N. K. Nagwani, and U. R. BP, “An optimized recommendation framework exploiting textual review based opinion mining for generating pleasantly surprising, novel yet relevant recommendations,” *Pattern Recognition Letters*, vol. 159, pp. 91–99, 2022.

[30] J. Li, K. Lu, Z. Huang, and H. T. Shen, “Two birds one stone: On both cold-start and long-tail recommendation,” in *Proceedings of the 2017 ACM on Multimedia Conference*, 2017, pp. 898–906.

[31] D. Chae, J. Kang, S. Kim, and J. Choi, “Rating augmentation with generative adversarial networks towards accurate collaborative filtering,” in *Proceedings of the World Wide Web Conference*, 2019, pp. 2616–2622.

[32] Z. Lin, W. Pan, and Z. Ming, “FR-FMSS: Federated recommendation via fake marks and secret sharing,” in *Proceedings of the 15th ACM Conference on Recommender Systems*, 2021, pp. 668–673.