ICMT: Item Cluster-Wise Multi-Objective Training for Long-Tail Recommendation

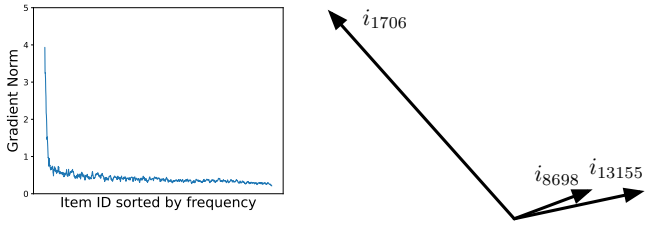
Yule Wang, Xin Xin, Yue Ding, Yunzhe Li, Yuxiang Shi, Dong Wang

***Abstract*—Item recommendation based on historical user-item interactions is of vital importance for web-based services. However, the data used to train a recommender system (RS) suffers from severe popularity bias. The item frequency distribution is a highly skewed long-tail distribution. Interactions of a small fraction of popular (head) items account for almost the whole training data. Normal training methods from such biased data tend to repetitively generate recommendations from the head items, which further exacerbates the data bias and affects the exploration of potentially interesting items from niche (tail) items.**

In this paper, we explore the central theme of long-tail recommendation. Through an empirical study, we find that head items are very likely to be recommended due to the fact that *the gradients coming from head items dominate the overall gradient update process, which further affects the optimization of tail items.* To this end, we propose a general framework namely *Item Cluster-Wise Multi-Objective Training* (ICMT) for long-tail recommendation. Firstly, the disentangled representation learning is utilized to identify the popularity impact behind user-item interactions. Then item clusters are adaptively formulated according to the disentangled popularity representation. After that, we consider the learning over the whole training data as a weighted aggregation of multiple (item) cluster-wise objectives, which can be resolved through a Pareto-Efficient solver for a harmonious overall gradient direction. Besides, a contractive loss focusing on model robustness is derived as a regularization term. We instantiate ICMT with three state-of-the-art recommendation models and conduct experiments on three real-world datasets. Experimental results demonstrate that the proposed ICMT significantly improves the overall recommendation performance, especially on tail items. Codes will be open-sourced.

I. Introduction

Recommender systems are emerging as a crucial role in online services and platforms to address the problem of information overload [1]–[3]. A recommender system (RS) is trained using historical user-item interactions with the target of providing the most interesting items given the current user state. However, there is a self-loop in the training of a RS [4]. The exposure mechanism of the RS affects the collection of user-item interactions, which are then circled back as the training data for the RS itself. Such self-loop leads to severe popularity bias in the training data. Specifically, the item frequency distribution in the training data is an extreme long-tail distribution [5]. A small fraction of popular (head) items accounts for almost the whole training dataset. Normal learning-to-rank methods [6] based on such biased data would lead to a situation that head items are pushed towards much higher ranking scores compared with other items. As a result, popular items are repetitively recommended, which further intensifies the popularity bias and the "rich get richer" Matthew effect [4].



(a) Gradient Norm Distribution (b) Gradient Conflict

Fig. 1: Empirical study on Gowalla.

Nevertheless, the recommendation from tail items plays an important role in improving the system performance. From the user's perspective, he/she could be easily bored with the repetitive popular recommendation. There are potentially relevant items that will lead to larger user satisfaction among tail items [7], [8]. For service providers, the recommendation from tail items can embrace more marginal profit compared with head items [9]. Generally speaking, the recommendation task is a typical exploitation-exploration problem. Long-tail recommendation will benefit both users and service providers with better exploration, which finally turns into larger profits in the long-run [10].

Existing methods that focus on long-tail recommendation are usually based on metrics like recommendation diversity and novelty [11]–[13]. However, these metrics are infeasible to directly optimize. Recommendations based on promoting such metrics could lead to a huge sacrifice of accuracy [14]. Besides, the definition of diversity or novelty is still an open research problem without a standard benchmark [15].

In this paper, we analyze the popularity bias problem of RS from an optimization perspective. We conduct an empirical study on the Gowalla[1] dataset. We train the state-of-the-art LightGCN [16] model on this dataset. Figure 1(a) visualizes the norm (i.e., L_2 norm $||\cdot||_2$) of gradients coming from different items. Figure 1(b) shows the gradients from a popular item i_{1706} and two tail items i_{8698}, i_{13155} in the dataset. We can have the following observation:

- Head items have much larger gradient norm than tail items, indicating that the overall gradient direction is actually dominated by head items.
- There are potential conflicts between gradients coming from head items and tail items. That is to say, updating model parameters based on gradients dominated by head items could potentially scarify the learning of tail items.

[1]https://snap.stanford.edu/data/loc-gowalla.html

Motivated from the above observation, we propose *Item Cluster-Wise Multi-Objective Training* (ICMT) for long-tail recommendation to address the popularity bias. Note that ICMT is a general learning framework and can be instantiated with different specific models, such as Probabilistic Matrix Factorization (PMF) [17], NeuMF [3], etc. More precisely, in the first place, a universal popularity embedding is involved in the ranking score prediction. This popularity embedding is then disentangled from user interest embedding for the modeling of popularity impact. Based on the disentangled representations, we split items into different clusters according to their correlation with the popularity embedding. We then consider the learning on each item cluster as an optimization objective. As a result, the learning over the whole training data can be seen as a weighted aggregation of multiple cluster-wise objectives. Then we utilize a Pareto-Efficient (PE) solver to adaptively learn the weight of each objective. Through the PE solver, we can find a solution that every cluster-wise objective is optimized without hurting the other one. In other words, the learning of head items would not affect the learning of tail items. Finally, a contractive loss focusing on model robustness is introduced as a regularization term to further prevent the potential overfitting of head items.

To summarize, this work makes the following contributions:

- We propose to tackle the long-tail recommendation task from an item cluster-wise optimization perspective. We show that head items are high likely to be recommended due to the domination of gradients, providing new directions to address the popularity bias of RS.
- We propose a general long-tail recommendation framework ICMT which is featured with popularity disentanglement, cluster-wise multi-objective optimization, and robust contractive regularization.
- We instantiate ICMT with three state-of-the-art recommendation models and conduct experiments on three real-world datasets. Experimental results demonstrate that ICMT significantly alleviates the popularity bias problem in recommender systems.

II. RELATED WORK

A. Methods for long-tail recommendation

Due to the popularity bias and the exposure mechanism of RS, tail items usually have much less training data. As a result, generating recommendations from head items is a conservative but effective way to improve recommendation accuracy (e.g., recall) [8]. To get rid of the conformity influence, some existing methods argue that other metrics like diversity [18]–[20] and novelty [13], [21] should be considered simultaneously as an additional regularization term. For example, [14] proposed a metric based on unpopularity of items. [13], [21], [22] consider diversity as the item difference within one recommendation list while novelty as the difference across lists. However, this kind of metrics is usually infeasible to directly optimize. Also, [23], [24] utilized knowledge transfer from many-shot head items to enhance the quality of tail-item embeddings. Inverse-propensity-scoring (IPS) is a practical one for industry product [25]. Since it is relatively easy to reweight training samples and ameliorate the distribution shift problem. Nevertheless, it suffers severely from high variance.

Besides, there are also methods based on additional knowledge input such as side-information, user feedback, and niche item clustering to relieve the cold start problem of tail items [26], [27]. However, none of the above work emphasized easing the neglect of tail items during the gradient update process.

B. Multi-Objective Optimization in RS

Despite recommendation accuracy as the main objective of recommendation, some researches have also been done focusing on other objectives such as availability, profitability, and usefulness [28], [29]. Besides, metrics about long-tail recommendation such as diversity and novelty are also considered as objectives [13], [21]. Recently, user-oriented objectives such as user sentiment are considered for better recommendation [30], [31]. For a commercial RS, CTR (Click Through Rate) and GMV (Gross Merchandise Volume) are included in [32], [33] to gain higher profits.

The optimization methods for multiple objectives can be categorized into two categories: heuristic search [34] and scalarization [35], [36]. Evolutionary algorithms are popular choices for heuristic search which deal simultaneously with a series of possible solutions in a single run [14]. But these algorithms usually depend heavily on the heuristic experience [34], [37]. Scalarization methods transform multiple objectives into a single one with a weighted sum of all objective functions [36]. Then the overall objective function is optimized to be Pareto-Efficient, where no single objective can be further improved without hurting the others [33].

In this paper, we aim to address the long-tail recommendation task from an optimization perspective. Unlike existing methods which introduce new metrics as the objectives [13], [21], [33], we consider the learning on a cluster of items as an objective. After that, we focus on finding an optimal solution in which the learning over one item cluster would not affect the learning on the other one.

III. METHODOLOGIES

In this section, we describe details of the proposed ICMT framework. Figure 2 illustrates the structure of ICMT, which contains a base recommendation model with the structural disentanglement of popularity, a PE solver to find the non-conflict gradient directions for multiple item cluster-wise objectives and a hybrid training loss consisting of weighted binary cross-entropy (BCE) and robust contractive regularization.

A. Base Model with Popularity Disentanglement

A traditional recommender base model mainly consists of user side f_u and item side f_i. For each user-item pair (u, i), the core task is to map user u and item i into a user interest embedding $\boldsymbol{v}_u \in R^{1\times D}$ and an item embedding $\boldsymbol{v}_i \in R^{1\times D}$

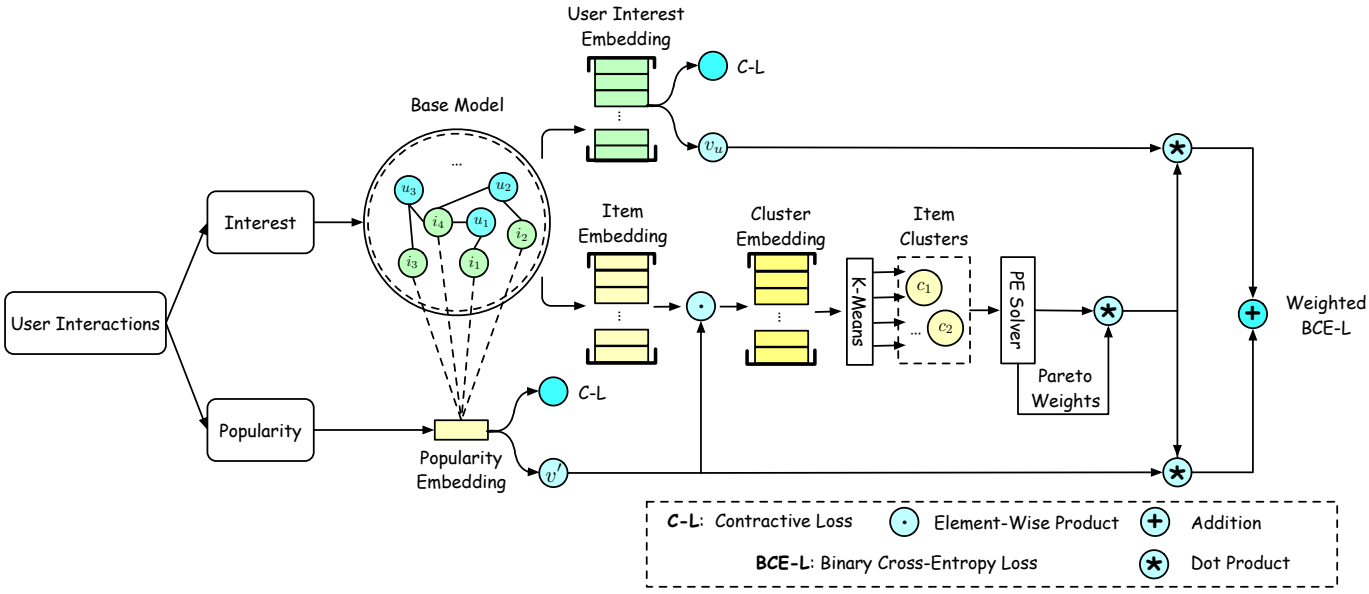


Fig. 2: An holistic illustration of ICMT. The reasons behind user-item interactions are firstly disentangled into popularity and interest factors. Then, interest factors are modeled through a base recommendation model (e.g. PMF, NeuMF, LightGCN), in which the user interest embeddings and item embeddings are obtained. Thereafter, items are clustered according to the correlation between item embeddings and the popularity embedding. Then the training objective is formulated as a weighted sum of cluster-wise objectives, which is optimized by a PE solver. Finally, a robust contractive loss is added as regularization.

respectively, where D denotes the embedding size. It can be formulated as:

$$\begin{aligned} \boldsymbol{v}_u &= f_u(u), \\ \boldsymbol{v}_i &= f_i(i). \end{aligned} \tag{1}$$

In traditional base models, when user and item embeddings are obtained, a scoring function g_b is utilized to calculate the relevance score for a user and an item(e.g., inner product [2], [16]), which can be formulated as:

$$\hat{y}_{ui}^{tra} = g_b(\boldsymbol{v}_u, \boldsymbol{v}_i). \tag{2}$$

Recently, many works design novel f_u and f_i to extract better features. For example, neural matrix factorization (NeuralMF) [3] adopts MLP as f_u and f_i to extract user and item features respectively, LightGCN [16] introduces the graph convolutional mechanism on f_u and f_i. However, the engagement of popularity impact is ignored by this approach, where different reasons of an interaction are bundled together as unified user representations. Therefore, we separate representations for user interest and popularity through assigning each user a unique interest embedding $\boldsymbol{v}_u$ while maintaining a general popularity embedding $\boldsymbol{v'} \in R^{1\times D}$. The latter one represents the public preference that participates with the item embedding $\boldsymbol{v}_i$ in every interaction, which is denoted as function g_p. Through aggregation, the prediction function of ICMT can be formulated as:

$$\hat{y}_{ui}^{ICMT} = \underbrace{g_b(\boldsymbol{v}_u, \boldsymbol{v}_i)}_{inference} + \lambda_p g_p(\boldsymbol{v'}, \boldsymbol{v}_i). \tag{3}$$

where λ_p is a weighting parameter controlling the ratio of popularity impact. During inference stage, only the interest part g_b will be used for final recommendation, where the popularity embedding has been disentangled. Without loss of generality, we implement g_p as matrix factorization in ICMT.

B. Item Cluster-Wise Pareto Efficiency

In this paper, we consider training the RS from implicit feedback (i.e., observed interactions are considered as positive samples while negative samples are sampled from missing interactions). As for the base model extracting user interest and item representations, we start from the normal training setting, in which each training sample has equal weight (i.e., 1). The total loss function is defined on the whole training data, which is shown as

$$\min_{\theta} L(\theta) = \sum_{(u,i)\in\mathcal{R}^+} l_{ui}(\theta) + \sum_{(u,i)\in\mathcal{R}^-} l_{ui}(\theta), \tag{4}$$

where $\mathcal{R}^+$ and $\mathcal{R}^-$ denote the set of positive samples and the set of sampled negative samples, correspondingly. l_{ui} is the specific loss (i.e., cross-entropy) of (u, i) pair and θ denotes base model parameters. Then we expand the loss on positive examples[2] as

$$L^+(\theta) = \sum_{(u,i)\in\mathcal{R}^+} l_{ui}(\theta) = \sum_{i\in\mathcal{I}} \sum_{u\in\mathcal{R}_i^+} l_{ui}(\theta), \tag{5}$$

where $\mathcal{I}$ denotes the whole item set and $\mathcal{R}_i^+$ is the set of users who have interacted with item i.

[2]We don't consider the loss on negative examples since we assume that they are sampled uniformly.

Due to the popularity bias in the training data, $\mathcal{R}_i^+$ is extremely imbalanced. For head items, $\mathcal{R}_i^+$ contains much more samples than tail items. As a result, when we perform updates according to $\partial L/\partial\theta$, the majority of gradients would come from the loss on head items. When there are conflicts between gradients coming from head items and gradients coming from tail items, the normal training setting would scarify the learning on tail items to achieve a lower overall loss. That is to say, the overall gradient direction is actually dominated by head items.

To address this problem, we propose to consider the learning on each item as an objective and formulate the training over whole (positive) data as a weighted sum of these multiple item-wise objectives:

$$\sum_{i\in\mathcal{I}} w_i \sum_{u\in\mathcal{R}_i^+} l_{ui}(\theta) = \sum_{i\in\mathcal{I}} w_i L_i^+(\theta). \tag{6}$$

$L_i^+(\theta)$ can be seen as the initial item-wise optimization objective. w_i is the weight for objective i.

We aim to find a set of w_i so that each item-wise objective can be optimized without hurting the others. In the real-world scenario, there are usually a huge amount of items. As a result, naively consider each $L_i^+(\theta)$ as an optimization objective could heavily increase the computation complexity of finding w_i. To address this problem, we first split the whole item set to K clusters and then consider the learning on each cluster of items.

Besides, in the total learning space θ, there are some parameters θ_{sh} which are shared between objectives and some parameters θ_i which are only related to item i. We just need to consider θ_{sh} for the selection of objective weights, since θ_i would not affect the learning of other items. It's worth mentioning that though $\boldsymbol{v}'$ is a shared embedding among between all the objectives, its purpose is to evenly capture the popularity preference among all the items. Therefore, ICMT doesn't apply the re-weighting strategy on $\boldsymbol{v}'$ and keeps it in the parameter set θ_i. Taking all the above factors, we formulate our loss function to train θ_{sh} on positive samples as

$$\begin{aligned} &L_{sh}^+(\theta) = \sum_{k=1}^{K} \omega_k L_k(\theta), \text{ where } L_k(\theta) = \sum_{i\in\mathcal{I}_k} L_i^+(\theta) \\ &s.t. \sum_{k=1}^{K} \omega_k = K, \omega_k \geq 0,\ \forall k = 1, ..., K \end{aligned} \tag{7}$$

where ω_k is the weight for the k-th item cluster and $\mathcal{I}_k$ denotes the set of items in this cluster. When $K = 1$, we can recover the normal training weight setting. L_k is the learning objective for cluster k. For the training of item-specific parameters θ_i, we still use the $L^+(\theta)$ as shown in Eq.(5).

In the following, we will first describe the item clustering strategy and then describe how to find ω_k for each cluster objective.

1) Adaptive Item Clustering: We cluster the items according to the observation that *items whose embeddings are similar with the popularity embedding should be separated out from items whose embeddings are dissimilar with popular embedding to formulate different item clusters*. We implement this idea through operating element-wise product on each $\boldsymbol{v}_i$ with $\boldsymbol{v}'$. The final item clustering embedding $\boldsymbol{v}_i^c$ is defined as follows:

$$\boldsymbol{v}_i^c = \frac{\boldsymbol{v}_i \odot \boldsymbol{v}'}{\|\boldsymbol{v}_i\| \cdot \|\boldsymbol{v}'\|}, \tag{8}$$

where $\odot$ denotes the element-wise product and this product is divided by the inner product (i.e., $\|\boldsymbol{v}_i\| \cdot \|\boldsymbol{v}'\|$) to normalize the scale impact. The clustering embedding mainly correlates with the item's similarity with the popularity embedding in terms of direction. Then, we perform K-Means clustering algorithm [38] using this item clustering embedding $\boldsymbol{v}_i^c$.

2) Pareto-Efficient Solver: In the following, we will describe how to find ω_k for each cluster objective. Firstly, we provide a brief introduction to Pareto-Efficiency and some related concepts.

Given a system that aims to minimize a series of objective functions $L_1, ..., L_K$, Pareto-Efficiency is a state when it is impossible to improve one objective without hurting other objectives. Formally, we provide the following definition:

Definition 1: For a minimization task of multiple objectives, let s_m and s_n denote two solutions as $s_m = (f_1^m, ..., f_K^m)$ and $s_n = (f_1^n, ..., f_K^n)$, s_m **dominates** s_n if and only if $f_1^m \leq f_1^n$, $\ldots$, $f_K^m \leq f_K^n$.

Then the concept of Pareto-Efficiency is defined as:

Definition 2: A solution $s = (f_1, ..., f_K)$ is **Pareto-Efficient** if and only if there is no other solution that dominates s .

It is worth mentioning that Pareto-Efficient solutions are not unique and the set of all such solutions is named as "Pareto Frontier". In this paper, we aim to find ω_k so that the solution of each cluster-wise objective is Pareto-Efficient, aka *Item Cluster-Wise Pareto-Efficiency.*

According to the definition of Pareto-Efficiency, we can use the Karush-Kuhn-Tucker (KKT) conditions [39] to describe the property of ω_k:

- $\omega_1, ..., \omega_K \geq 0$ and $\sum_{k=1}^K \omega_k = K$.
- For the shared parameters θ_{sh}: $\sum_{k=1}^K \omega_k \nabla_{\theta_{sh}} L_k(\theta) = 0$.

As a result, the task of finding ω_k can be formulated as

$$\begin{aligned} &\min_{\omega_1,...,\omega_K} \left\| \sum_{k=1}^{K} \omega_k \nabla_{\theta_{sh}} L_k(\theta) \right\|_2 \\ &s.t. \sum_{k=1}^{K} \omega_k = K, \omega_k \geq 0,\ \forall k = 1, ..., K \end{aligned} \tag{9}$$

The optimization problem defined in Eq.(9) is equivalent to finding a minimum-norm point in the convex hull of the set of input points (i.e., ω_k), which has been extensively studied [40].

Given the current L_k in one training step, we utilize the Frank-Wolfe algorithm [41] to solve the convex optimization problem of Eq.(9). Algorithm 1 shows the detail of this process. Its time complexity is mostly determined by the number of objectives and iterations with a time complexity upper bound of $O(K|d_p|)$, where $|d_p|$ is the dimension of parameter space θ_{sh}. Usually, the number of objectives is

limited, therefore the running time of Algorithm 1 is negligible compared to the model training time cost.

Algorithm 1 PE-solver

Input: θ, K, L_k
Output: $\boldsymbol{\omega}$: $[\omega_1, ..., \omega_K]$
initialize $\omega_1, ..., \omega_K = (1, ..., 1)$
compute $M_{ij} := (\nabla_{\theta_{sh}} L_i(\theta))^T \cdot \nabla_{\theta_{sh}} L_j(\theta)$
repeat
  $t := argmin_i \sum_{j=1}^{K} \omega_j M_{ij}$
  $\hat{\gamma} := argmin_\gamma((1-\gamma)\boldsymbol{\omega} + \gamma \boldsymbol{e}_t)^T \boldsymbol{M}((1-\gamma)\boldsymbol{\omega} + \gamma \boldsymbol{e}_t)$, where $\boldsymbol{e}_t$ is a vector whose elements are 1 at index t and 0 otherwise.
  $\boldsymbol{\omega} = (1-\hat{\gamma})\boldsymbol{\omega} + \hat{\gamma}\boldsymbol{e}_t$
until $\hat{\gamma} \sim 0$ **or** Number of Iterations Limit
return $\boldsymbol{\omega}$

C. Robust Contractive Regularization

With head items dominating the gradient update process, it has been demonstrated that many state-of-the-art recommendation models are actually fragile and vulnerable to small fluctuations and changes from head items [42]. In this section, we propose a simple yet effective penalty term to encourage model robustness and further downgrade the side impact from popularity bias.

For a robust recommendation model, the user representation $\boldsymbol{v}_u$ should remain a small change when there is a tiny fluctuation of item-specific parameter θ_i. Meanwhile, to refrain from the popularity impact, item representation should also be less disturbed by the popularity embedding parameter θ'. To this end, we define the contractive loss as the sum of the squared Jacobian matrix of $\boldsymbol{v}_u$ with respect to θ_i and $\boldsymbol{v}_i$ with respect to θ':

$$L_{con} = \sum_{i \in \mathcal{I}} \left(\sum_{u \in \mathcal{U}} \| \frac{\partial \boldsymbol{v}_u}{\partial \theta_i} \|^2 + \| \frac{\partial \boldsymbol{v}_i}{\partial \theta'} \|^2 \right). \tag{10}$$

This contractive loss is added as a regularization term to encourage more robust model training.

D. Training Details

In this paper, we use the BCE loss as the basement loss to train the recommendation model. More precisely, the specific loss l_{ui} for a (u, i) pair is formulated as

$$l_{ui}(\theta) = -[Y \log(\delta(\hat{y}_{ui}^{ICMT})) + (1-Y)\log(1-\delta(\hat{y}_{ui}^{ICMT}))], \tag{11}$$

where Y is the label for the (u, i) pair (i.e., $Y = 1$ if there is an interaction between user u and item i, otherwise $Y = 0$). δ is the sigmoid function. $\hat{y}_{ui}^{ICMT}$ is calculated according to Eq.(3).

Considering the robust contractive regularization, the final training function of ICMT regarding the shared parameter θ_{sh} is formulated as

$$L_{ICMT}^{sh} = L_{sh}^{+}(\theta) + \sum_{(u,i) \in \mathcal{R}^-} l_{ui}(\theta) + \lambda_c L_{con} + \lambda_1 \|\theta\|_2 . \tag{12}$$

Algorithm 2 Training and inference of ICMT

Input: $\mathcal{R}^+$,$\mathcal{R}^-$, recommendation model, K item clusters, learning rate η and all other hyperparameters
Output: parameters in the whole learning space θ: $\{\theta_{sh}, \theta_i\}$
initialize θ
sample a mini-batch of (u, i) from $\mathcal{R}^+$ and $\mathcal{R}^-$
for each batch **do**
  compute $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ through base recommendation model
  compute $\hat{y}_{ui}^{ICMT}$ according to Eq.(3)
  compute $\boldsymbol{v}_i^c$ according to Eq.(8).
  generating K item clusters utilizing K-Means algorithm according to $\boldsymbol{v}_i^c$.
  run Alg. 1 to update $\omega_1, ..., \omega_K$ = PEsolver(θ, L_k);
  compute L_{ICMT}^{sh} according to Eq.(12);
  compute L_{ICMT}^{i} according to Eq.(13);
  $\theta_{sh} \leftarrow \theta_{sh} - \eta \cdot \partial L_{ICMT}^{sh} / \partial \theta_{sh}$;
  $\theta_i \leftarrow \theta_i - \eta \cdot \partial L_{ICMT}^{i} / \partial \theta_i$;
end for
until converge
return θ
generate recommendation according to $g_b(\boldsymbol{v}_u, \boldsymbol{v}_i)$

where $L_{sh}^{+}(\theta)$ is calculated according to Eq.(7), λ_c and λ_1 are regularization coefficients. For item-specific parameter θ_i, the training function of ICMT is formulated as

$$L_{ICMT}^{i} = L^{+}(\theta) + \sum_{(u,i) \in \mathcal{R}^-} l_{ui}(\theta) + \lambda_c L_{con} + \lambda_1 \|\theta\|_2 . \tag{13}$$

where $L^{+}(\theta)$ is calculated according to Eq.(5).

In each training step of ICMT, the popularity impact, unique user interests, and items are firstly mapped into latent representations $\boldsymbol{v}'$, $\boldsymbol{v}_u$, and $\boldsymbol{v}_i$ by the recommendation model. The items are clustered through the K-Means algorithm according to their clustering embedding in the form of Eq.(8). We then run the PE-solver according to Algorithm 1 to get the weight ω_k for item cluster k. Then, we calculate the prediction score according to Eq.(3). For the shared parameter θ_{sh}, we perform update by minimizing L_{ICMT}^{sh} while for item-specific parameter θ_i, we perform update through minimizing L_{ICMT}^{i}. Algorithm 2 illustrates the overall training and inference procedure of ICMT.

IV. Experimental Setup

In this section, we conduct experiments aiming to answer the following research questions:

RQ1: How does the proposed IICMT perform compared with normal training and other long-tail recommendation algorithms?

RQ2: How do the components and hyper-parameters of ICMT affect the recommendation performance?

RQ3: Can the PE-solver find Pareto-Efficiency solutions for multiple item cluster-wise objectives?

RQ4: How is the interpretability of ICMT?

TABLE I: Dataset statistics.

DataSet	Last.Fm	Gowalla	Yelp2018
Users	1270	15612	31646
Items	4475	13701	21098
Interactions	156882	546299	1331183
Density	0.0276	0.00255	0.00199

A. Experimental Settings

1) Datasets: We conduct experiments on three public accessible datasets: Last.Fm [3], Gowalla and Yelp2018 [4]. The datasets vary in domains, platforms, and sparsity. Table I summarizes the statistics of the three datasets.

Last.Fm: This is a widely used dataset which contains 1 million ratings between users and movies. We binarize the ratings into implicit feedback. Interacted items are considered as positive samples. Due to the sparsity of the dataset, we use the 10-core setting, i.e., retaining users and items which have at least ten interactions.

Gowalla: This is the check-in dataset obtained from Gowalla, where users share their locations by checking-in behavior [43]. To ensure the quality of the dataset, we use the 20-core setting.

Yelp2018: This dataset is adopted from the 2018 edition of the Yelp challenge. Wherein, the local businesses like restaurants and bars are viewed as the items. Similarly, we use the 20-core setting to ensure that each user and item have at least twenty interactions.

2) Evaluation protocols.: We adopt cross-validation to evaluate the performance. The ratio of training, validation, and test set is 8:1:1. The ranking is performed among the whole item set. Each experiment is repeated 5 times and the average performance is reported.

The recommendation quality is measured both in terms of overall accuracy and long-tail performance that reflecting the alleviation of popularity bias. The overall accuracy is measured with two metrics: Recall and Normalized Discounted Cumulative Gain (NDCG). Recall@N measures how many ground-truth items are included in the top-N positions of the recommendation list. NDCG is a rank-sensitive metric that assigns higher weights to top positions in the recommendation list [44].

For the evaluation of long-tail performance, we first split items into $\mathcal{I}_h$ and $\mathcal{I}_t$ according to the ratio of 20% $\setminus$ 80% **Pareto Principle**. $\mathcal{I}_h$ represents the set of head items and $\mathcal{I}_t$ denotes the set of tail items. Here 20% means 20% of total item numbers, other than 20% of interactions. We then adopt the following four metrics.

Recall-Tail and NDCG-Tail: Recall-Tail@N measures how many tail items belong to $\mathcal{I}_t$ are included in the top-N positions of the recommendation list and then interacted by the user. Similarly, NDCG-Tail@N assigns higher weights to top positions.

Coverage and APT: Coverage measures how many different items appear in the top-N recommendation list. A more readily interpretable but closely related metric of success we will use for evaluation is the Average Percentage of Tail items (APT) in the recommendation lists. More precisely, Coverage@N and APT@N are defined as:

$$Coverage@N = \frac{|\cup_{u \in test} list_N(u)|}{|\mathcal{I}|} \tag{14}$$

$$APT@N = \frac{1}{|\mathcal{U}|} \sum_{u \in test} \frac{|list_N(u) \cap \mathcal{I}_t|}{|list_N(u)|} \tag{15}$$

where $|\mathcal{U}|$ and $|\mathcal{I}|$ are the number of users and items in the test set, $list_N(u)$ represents the list of top-N recommended items for each user u in test set.

3) Baselines: We instantiate the proposed ICMT with three state-of-the-art recommendation models:

- PMF [2]: Probabilistic Matrix Factorization models the conditional probability of latent factors given the observed ratings and includes Gaussian priors as regularization.
- NeuMF [3]: Neural Matrix Factorization is one notable deep learning-based recommendation model. It combines matrix factorization and multi-layer perceptrons (MLP) to learn high-order interaction signals.
- LightGCN [16]: LightGCN is a graph-based model that learns user and item representations by linearly propagating them on the interaction graph. The user and item embedding is formulated as the aggregation of hidden vectors in all layers.

Each model is trained with the following model-agnostic frameworks:

- Normal Training: This is the normal training procedure with simple BCE loss, as shown in Eq.(1).
- IPS [45]: IPS re-weights each interaction according to item popularity. Specifically, weight for an interaction is set as the inverse of corresponding item popularity value.
- PO-EA [13]: PO-EA utilizes an evolutionary algorithm to find Pareto-Efficient solutions for multiple objectives like accuracy, diversity, and novelty.
- MORS [14]: MORS [14] proposes a novel multi-objective evolutionary algorithm to find trade-off solutions between recommending accurate and niche items simultaneously.
- FA-reg [46]: Fairness-aware regularization (FA-reg) introduces a flexible regularization-based framework to enhance the long-tail coverage of recommendation lists in a learning-to-rank algorithm.
- ICMT: our proposed learning framework.

4) Parameter Settings.: All methods are learned with the Adam optimizer [47] except utilizing RMSprop optimizer in NeuMF based models. The batch size is set as 512. The learning rate is set as $1e^{-3}$. We evaluate on the validation set every 3000 batches of updates. For a fair comparison, the embedding size is set as 64 for all models. For NeuMF and LightGCN, we utilize a three-layer-structure. The node-dropout and message-dropout in LightGCN are set as 0.1 on all datasets. For hyperparameters of ICMT, λ_p, λ_c, and λ_1 are searched between {1e-4, 1e-3, 2e-3, 5e-3, 1e-2} on all three datasets. We set item cluster number $K = 2$ without special

[3]https://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip

[4]https://www.kaggle.com/yelp-dataset/yelp-dataset/version/7

TABLE II: Top-20 long-tail performance on three datasets. RT, NT, Cov and AT are short for Recall-Tail, NDCG-Tail, Coverage and APT respectively. Boldface denotes the highest score.

		Last.Fm				Gowalla				Yelp2018			
Base	Methods	RT@20	NT@20	Cov@20	AT@20	RT@20	NT@20	Cov@20	AT@20	RT@20	NT@20	Cov@20	AT@20
PMF	Normal	0.0014	0.0011	0.3692	0.1252	0.0399	0.0193	0.3428	0.1140	0.0048	0.0024	0.3286	0.0611
	IPS	0.0009	0.0009	0.3034	0.0713	0.0292	0.0154	0.3064	0.0926	0.0029	0.0015	0.2845	0.0387
	PO-EA	0.0011	0.0009	0.3106	0.0863	0.0337	0.0171	0.3293	0.1068	0.0039	0.0028	0.2992	0.0543
	MORS	0.0015	0.0012	0.3714	0.1246	0.0420	0.0203	0.3640	0.1233	0.0055	0.0030	0.3437	0.0738
	FA-reg	0.0016	0.0012	0.3914	0.1373	0.0418	0.0203	0.3722	0.1332	0.0056	0.0031	0.3512	0.0745
	ICMT	**0.0019***	**0.0014***	**0.4785***	**0.1542***	**0.0448***	**0.0226***	**0.4112***	**0.1451***	**0.0070***	**0.0036***	**0.3818***	**0.0862***
NeuMF	Normal	0.0023	0.0025	0.6235	0.2336	0.0249	0.0123	0.5908	0.2166	0.0052	0.0028	0.5617	0.0969
	IPS	0.0018	0.0020	0.5580	0.1778	0.0197	0.0099	0.5053	0.1531	0.0032	0.0021	0.4740	0.0695
	PO-EA	0.0021	0.0022	0.5853	0.1911	0.0215	0.0121	0.5392	0.2086	0.0044	0.0025	0.5273	0.0745
	MORS	0.0031	0.0031	0.6188	0.2447	0.0284	0.0144	0.6477	0.2589	0.0061	0.0029	0.5897	0.1089
	FA-reg	0.0029	0.0030	0.6242	0.2429	0.0293	0.0133	0.6581	0.2650	0.0062	0.0029	0.5875	0.1081
	ICMT	**0.0054***	**0.0032***	**0.6418***	**0.2770***	**0.0348***	**0.0171***	**0.6724***	**0.3188***	**0.0067***	**0.0030***	**0.6350***	**0.1339***
LGC	Normal	0.0034	0.0025	0.4273	0.1835	0.0425	0.0197	0.4195	0.1257	0.0071	0.0035	0.3767	0.0708
	IPS	0.0022	0.0013	0.3508	0.1065	0.0364	0.0135	0.3547	0.0920	0.0042	0.0021	0.2955	0.0545
	PO-EA	0.0034	0.0027	0.4115	0.1790	0.0388	0.0160	0.3710	0.1178	0.0064	0.0029	0.3035	0.0639
	MORS	0.0049	0.0030	0.4248	0.1951	0.0433	0.0215	0.4291	0.1391	0.0094	0.0048	0.3923	0.0919
	FA-reg	0.0039	0.0026	0.4211	0.1807	0.0435	0.0219	0.4307	0.1331	0.0081	0.0043	0.3841	0.0923
	ICMT	**0.0057***	**0.0033***	**0.4418***	**0.2043***	**0.0497***	**0.0234***	**0.4860***	**0.1600***	**0.0106***	**0.0053***	**0.4677***	**0.1168***

$*$ denotes significance p-value <0.01 compared with normal training.

TABLE III: Top-20 overall performance on three datasets. R and NG are short for Recall and NDCG respectively. Boldface denotes the highest score.

		Last.Fm		Gowalla		Yelp2018	
Base	Methods	R@20	NG@20	R@20	NG@20	R@20	NG@20
PMF	Normal	0.0303	0.0309	0.1719	0.1463	0.0507	0.0377
	IPS	0.0278	0.0283	0.1306	0.1219	0.0382	0.0291
	PO-EA	0.0282	0.0292	0.1459	0.1304	0.0456	0.0345
	MORS	0.0301	0.0302	0.1614	0.1398	0.0480	0.0360
	FA-reg	0.0292	0.0298	0.1623	0.1385	0.0474	0.0361
	ICMT	**0.0329***	**0.0327***	**0.1754***	**0.1489***	**0.0513***	**0.0386***
NeuMF	Normal	0.0236	0.0229	0.1148	0.0920	0.0388	0.0280
	IPS	0.0207	0.0209	0.0865	0.0771	0.0302	0.0219
	PO-EA	0.0215	0.0207	0.0917	0.0848	0.0335	0.0234
	MORS	0.0228	0.0206	0.1066	0.0949	0.0388	0.0279
	FA-reg	0.0210	0.0208	0.1090	0.0918	0.0385	0.0283
	ICMT	**0.0237**	**0.0231***	**0.1237***	**0.0953***	**0.0416***	**0.0297***
LGC	Normal	0.0413	0.0391	0.1873	0.1616	0.0580	0.0437
	IPS	0.0396	0.0383	0.1494	0.1278	0.0423	0.0312
	PO-EA	0.0404	0.0382	0.1669	0.1439	0.0479	0.0369
	MORS	0.0407	0.0382	0.1719	0.1542	0.0546	0.0415
	FA-reg	0.0406	0.0386	0.1762	0.1544	0.0531	0.0415
	ICMT	**0.0432***	**0.0404***	**0.1898***	**0.1626***	**0.0599***	**0.0448***

$*$ denotes significance p-value <0.01 compared with normal training.

mention. Note that model hyperparameters keep exactly the same across all different training frameworks for a fair comparison.

B. Performance Comparison (RQ1)

Table II and Table III shows the performance of top-N recommendation on Last.Fm, Gowalla and Yelp2018, respectively. We make the following observations from the results:

(1). According to Table II, ICMT achieves the best long-tail recommendation performance among all methods. This observation confirms that the proposed ICMT is effective to alleviate the popularity bias and generate better recommendations from tail items. The base model with ICMT achieves the average absolute Recall-Tail@20 / NDCG-Tail@20 / Coverage@20 / APT@20 / gain of 42.45% / 30.11% / 15.45% / 33.23%.

(2). As shown in Table III, although ICMT is proposed to tackle the long-tail recommendation task, in all cases, it outperforms normal training and the other long-tail recommendation methods in terms of overall accuracy. It demonstrates that ICMT achieves a better trade-off between long-tail recommendation and overall accuracy compared with other frameworks. The performance gain regarding the overall accuracy metrics NDCG@20 / Recall@20 is 4.03 % / 2.98 %. This improvement mainly comes from promoting the high-quality niche items while downgrading the irrelevant head items.

(3). We conduct one-sample t-tests and the obtained results (i.e., p-value <0.01) indicate that the improvement regarding both long-tail recommendation metrics and overall metrics of ICMT is statistically significant.

(4). We also analyze the head items and tail items trade-off in ICMT. Figure 3 visualizes the average recommendation accuracy (i.e., NDCG@20, Recall@20) on head items and tail items on Gowalla dataset with LightGCN as the recommendation model. Generally, the points which fall into the top-right direction indicate better performance with both higher head and tail accuracy. It's obvious that the proposed ICMT achieves the highest tail accuracy while only scarifies a little head accuracy compared with normal training. However, other methods can not achieve such a performance. In most cases, these methods tend to lead to a larger decrease in head accuracy while obtaining a smaller gain on tail accuracy. This result demonstrates that ICMT achieves a better trade-off between head items and tail items, compared with other methods. The performance gain of ICMT mainly comes from the growth in niche items without the loss across head items.

To conclude, the proposed ICMT significantly improves the long-tail recommendation performance compared with normal training, and meanwhile, enhances the overall accuracy.

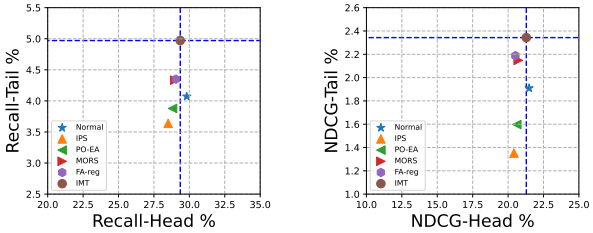


Fig. 3: NDCG@20 and Recall@20 Profile of head and tail items on Gowalla Dataset

TABLE IV: Ablation study of method components on three datasets. Boldface denotes highest scores. w/o denotes without. ↓ indicates as a severe performance drop (more than 10%).

Dataset	Methods	Long-Tail Metrics				Overall Metrics	
		RT@20	*NT*@20	*Cov*@20	*AT*@20	*R*@20	*NG*@20
Last.Fm	Default	**0.0037**	**0.0033**	**0.4418**	**0.2043**	**0.0432**	**0.0404**
	w/o CR	0.0034 ↓	0.0027 ↓	0.4340	0.1890	0.0421	0.0389
	w/o PD	0.0035	0.0030	0.4248	0.1829	0.0404	0.0381
	w/o CL	0.0036	0.0030	0.4373	0.1925	0.0412	0.0391
Gowalla	Default	**0.0497**	**0.0234**	**0.4860**	**0.1600**	**0.1898**	**0.1626**
	w/o CR	0.0443 ↓	0.0207 ↓	0.4677	0.1508	0.1885	0.1616
	w/o PD	0.0467	0.0213	0.4434 ↓	0.1428 ↓	0.1864	0.1597
	w/o CL	0.0495	0.0232	0.4768	0.1573	0.1885	0.1614
Yelp2018	Default	**0.0106**	**0.0053**	**0.4677**	**0.1168**	**0.0599**	**0.0448**
	w/o CR	0.0095 ↓	0.0046 ↓	0.4659	0.1012	0.0594	0.0444
	w/o PD	0.0097	0.0052	0.4411	0.0945 ↓	0.0590	0.0441
	w/o CL	0.0100	0.0049	0.4627	0.1119	0.0596	0.0447

C. Ablation Study and Hyper-parameter Study (RQ2)

1) Ablation Study: In this part, we conduct ablation study to analyze the functionality of the three components of ICMT (i.e.,cluster-wise re-weighting (CR), popularity disentanglement (PD), and contractive loss (CL)). Table IV shows the performance of ICMT and its variants on all three datasets utilizing LightGCN as the base recommendation model. We introduce the variants and analyze their effects respectively:

(1). *Remove Cluster-wise Re-weighting (w/o CR)*: The most significant long-tail accuracy degradation occurred without the re-weighting strategy, which implies that niche items tend to obtain higher weights because of their weak relationship with popularity embedding. This proves that clustering the items and treating the recommendation target as a multi-objective optimization problem can greatly improve the performance of the model in the long-tail space.

(2). *Remove Popularity Disentanglement (w/o PD)*: After taking away the popularity embedding and perform clustering on the item embedding distribution, we find that Coverage and APT are significantly worse, meaning that the vanilla user embedding is biased by the item popularity. On the other hand, with the participation of popularity embedding, user interest embedding in ICMT alone reflects the users' true preference, which can thus explore more niche items and improve the long-tail performance.

(3) *Remove Contractive Loss (w/o CL)*: We can see that the long-tail results downgrade without the contractive regularization, manifesting that the niche items are boosted after regularizing with contractive Jacobi gradients.

To sum up, the combination of these three strategies (i.e., ICMT) yields the best performance, proving that all the three components of ICMT are effective and work collaboratively to improve the long-tail recommendation performance and overall accuracy.

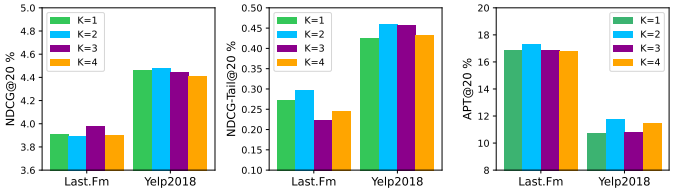


Fig. 4: Effect of item cluster number K.

2) Hyper-parameter Study:

(1). *Effect of Item Cluster Number K*: In this part, we use LightGCN as the base recommendation model since it has the best overall accuracy performance. Here we choose $K \in \{1, 2, 3, 4\}$ to conduct the experiment. Figure 4 illustrates, NDCG@20, NDCG-Tail@20, and APT@20 under different cluster numbers on Last.Fm and Yelp2018 dataset. The general observation is that overall recommendation accuracy maintains the same level while the long-tail performance shows bell-shaped curves. Increasing the cluster number from 1 (i.e., normal training) to 2 leads to the largest long-tail performance improvement. Later on, long-tail performance keeps diminishing along with the increase of cluster number. Such experimental results indicate that adaptively assigning the items into two clusters leads to the most satisfactory performance. However, with more item clusters, the performance of ICMT on long-tail items gets compromised. The reason could be that too many clusters would make ICMT put more focus on the balance between tail clusters rather than the balance between head and tail items.

(2). *Effect of Popularity Factor Weight λ_p*: To evaluate the impact of popularity factor weight λ_p, we vary the weight in the range of {0, 1e-4, 1e-3, 2e-3, 5e-3, 1e-2}. The experimental results are summarized in Figure 5. We can observe that the overall NDCG@20 reaches its peak when $\lambda_p = 2e-3$, thus demonstrating that properly disentangle the popularity factor closely reflects the true user interest. Meanwhile, with the incremental of λ_p, the long-tail performance keeps rising swiftly. This implies that more niche items are excavated when we put more emphasis on decomposing the popularity factor. To sum up, for a higher overall accuracy, we choose $\lambda_p = 2e-3$ as our default setting.

(3). *Effect of Contractive Loss Weight λ_c*: From Figure 6, we can observe that a small value of λ_c promotes the accuracy, especially in the long-tail space according to its ability of balancing the gradients from all items and suppressing the popularity embedding dominance. However, despite keep promoting the APT, a larger portion of robust regularization does not necessarily lead to better accuracy due to the issue of losing information from the gradients. Therefore, we set λ_c as (1e-3, 1e-3, 2e-3) corresponding to Last.Fm, Gowalla, Yelp2018 to achieve the best overall performance.

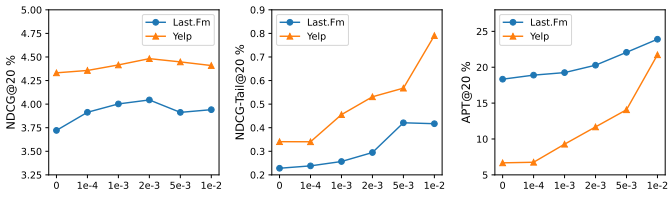


Fig. 5: Effect of Popularity Factor Weight λ_p.

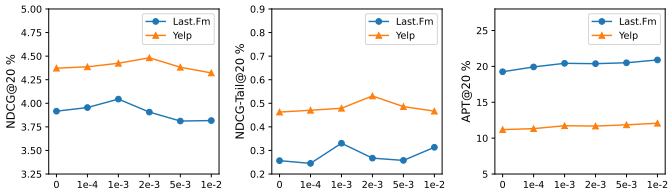


Fig. 6: Effect of Contractive Loss Weight λ_c.

D. PE-Solver Investigation (RQ3)

In this part, we conduct experiments to show whether the PE-solver generates reasonable Pareto-Efficient solutions.

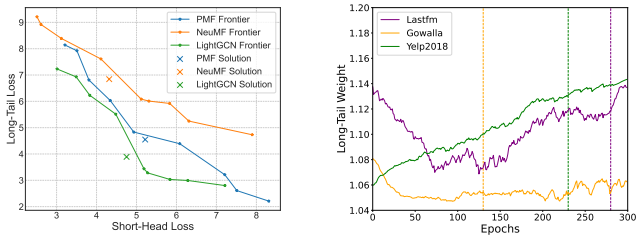
1) Pareto Frontier and the Searched PE Point: On the Gowalla dataset with all the three recommendation models, we first generate the Pareto Frontiers of head and tail losses by running the Pareto MTL algorithm [33] with different trade-off preference vectors, shown in Figure 7(a). It can be observed that the obtained Pareto Frontiers under different constraints follow Pareto-Efficiency, i.e., no point achieves both lower short-head and long-tail losses than other points. When the model focuses more on head items, the short-head loss is lower while the long-tail loss increases, and vice versa.

When it comes to the searched point of the PE solver, we can see that on all recommendations models, those points mainly lie in the middle part of the Pareto Frontiers. This observation indicates that the PE-solver coincides with our aim of balancing the trade-off between head items and tail items.

2) The Learning of Weights: To be clear of the training process and reveal the attention of long-tail items, we further plot the learning curves of the average weights assigned to long-tail items, as shown in Figure 7(b). We use LightGCN as the recommendation model and visualize the trend on all three datasets. We can observe that the obtained weights from PE-solver tend to focus on tail items. After variating at the early training stage, the weight for tail items becomes flattened and then converges to a value around [1.04, 1.16]. On the other hand, the normal training methods neglect these PE weights and treat all items in the same way, leading to the overfitting on head items. Hence, the proposed ICMT can effectively eliminate the popularity bias of RS by assigning adaptive weights to head and tail cluster-wise objectives.

E. Case Study (RQ4)

To show the interpretability of ICMT, in the Last.Fm dataset, we randomly select a user u_{715} and retrieve his



(a) The Pareto Frontier and searched solutions

(b) Long-tail weights learning curves. Vertical bars mark convergence epochs.

Fig. 7: PE-solver investigation

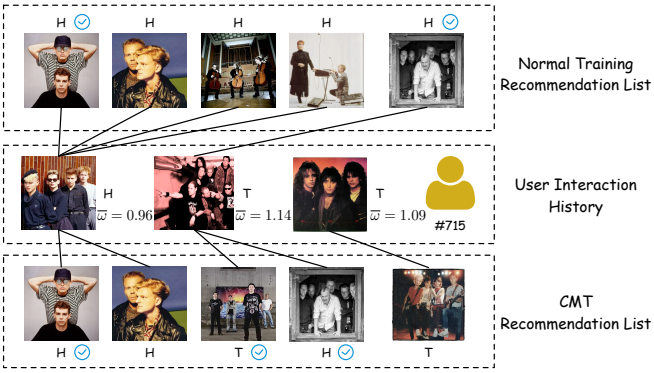


Fig. 8: Case study of a single user. The upper part is the recommendation list from LightGCN-Normal and the lower part is the recommendation list of LightGCN-ICMT. H denotes the movie is a head item while T denotes the movie is a tail item. $\overline{\omega}$ denotes the average weight assigned to each item. The blue check mark denotes that the recommended item belongs to the ground-truth items in the test set. The arrow indicates that there are some relations between the two items (e.g., same tag).

two top-5 recommendation lists from LightGCN-Normal and LightGCN-ICMT given the same interaction history. Figure 8 illustrates the recommendation detail. We observed that the recommendation list derived from LightGCN-Normal contains popular ground-truth items but does not contain any tail items. LightGCN-ICMT contains two tail (unpopular) items, *Farben Lehre* and *Plavi Orkestar* thanks to the higher average weights assigned to tail items in his interaction history. One of the two recommended tail items belongs to the ground-truth test set, which improves the long-tail and overall performance.

Note that the two recommendation lists have several items in common (e.g. *Erasure*), which indicates that LightGCN-ICMT can also catch the preference on popular items.

V. Conclusion

In this paper, we propose to tackle the long-tail recommendation task from a multi-objective optimization perspective. We find that head items are repetitively recommended due to the fact that head items tend to have larger gradient norms and

thus dominate the gradient updates. Learning parameters based on such gradients could scarify the performance of long-tail items. To alleviate such a phenomenon, we propose a general learning framework namely ICMT which is featured with popularity disentanglement, cluster-wise multi-objective re-weighting, and robust contractive regularization. We instantiate ICMT with three state-of-the-art recommendation models and conduct extensive experiments on three real-world datasets. The results demonstrate the effectiveness of ICMT. Future work includes generalizing ICMT for other tasks such as multi-class classification and long-tail document retrieval.

REFERENCES

[1] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.

[2] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, pp. 1257–1264, 2007.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[4] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," *arXiv preprint arXiv:2010.03240*, 2020.

[5] H. Abdollahpouri, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[7] 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 25th ACM international conference on Multimedia*, 2017, pp. 898–906.

[8] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896–911, 2011.

[9] C. Anderson, *The Long Tail*. Hachette Books, 2006.

[10] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac, "What recommenders recommend: an analysis of recommendation biases and possible countermeasures," *UMUI*, vol. 25, no. 5, pp. 427–491, 2015.

[11] Q. Wu, Y. Liu, C. Miao, B. Zhao, Y. Zhao, and L. Guan, "Pd-gan: Adversarial learning for personalized diversity-promoting recommendation." in *IJCAI*, vol. 19, 2019, pp. 3870–3876.

[12] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *PNAS*, vol. 107, no. 10, pp. 4511–4515, 2010.

[13] M. T. Ribeiro, N. Ziviani, E. S. D. Moura, I. Hata, A. Lacerda, and A. Veloso, "Multiobjective pareto-efficient approaches for recommender systems," *TIST*, vol. 5, no. 4, pp. 1–20, 2014.

[14] S. Wang, M. Gong, H. Li, and J. Yang, "Multi-objective optimization for long tail recommendation," *Knowledge-Based Systems*, vol. 104, pp. 145–155, 2016.

[15] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: evaluating recommender systems by coverage and serendipity," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 257–260.

[16] X. He, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *SIGIR*, 2020, pp. 639–648.

[17] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[18] M. Kaminskas and D. Bridge, "Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems," *TiiS*, vol. 7, no. 1, pp. 1–42, 2016.

[19] N. J. Hurley, "Personalised ranking with diversity," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 379–382.

[20] W. Chen, P. Ren, F. Cai, F. Sun, and M. de Rijke, "Improving end-to-end sequential recommendations with intent-aware diversification," in *CIKM*, 2020, pp. 175–184.

[21] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, "Pareto-efficient hybridization for multi-objective recommender systems," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 19–26.

[22] L. Shi, "Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 57–64.

[23] S. Jang, H. Lee, H. Cho, and S. Chung, "Cities: Contextual inference of tail-item embeddings for sequential recommendation," in *ICDM*. IEEE, 2020, pp. 202–211.

[24] Y. Zhang, D. Z. Cheng, T. Yao, X. Yi, L. Hong, and E. H. Chi, "A model of two tales: Dual transfer learning framework for improved long-tail item recommendation," in *WWW 2021*, 2021, pp. 2220–2231.

[25] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," *Advances in neural information processing systems*, vol. 19, pp. 601–608, 2006.

[26] B. Bai, Y. Fan, W. Tan, and J. Zhang, "Dltsr: A deep learning framework for recommendation of long-tail web services," *IEEE Transactions on Services Computing*, 2017.

[27] Y. Kim, K. Kim, C. Park, and H. Yu, "Sequential and diverse recommendation with long tail." in *IJCAI*, 2019, pp. 2740–2746.

[28] T. Jambor and J. Wang, "Optimizing multiple objectives in collaborative filtering," in *RecSys*, 2010, pp. 55–62.

[29] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *CHI'06*, 2006, pp. 1097–1101.

[30] C. Musto, M. de Gemmis, G. Semeraro, and P. Lops, "A multi-criteria recommender system exploiting aspect-based sentiment analysis of users' reviews," in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 321–325.

[31] M. Rodriguez, C. Posse, and E. Zhang, "Multiple objective optimization in recommender systems," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 11–18.

[32] P. Nguyen, J. Dines, and J. Krasnodebski, "A multi-objective learning to re-rank approach to optimize online marketplaces for multiple stakeholders," *arXiv preprint arXiv:1708.00651*, 2017.

[33] X. Lin, H. Chen, C. Pei, F. Sun, X. Xiao, H. Sun, Y. Zhang, W. Ou, and P. Jiang, "A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 20–28.

[34] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[35] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda)," 2009.

[36] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping, "Fairness-aware group recommendation with pareto-efficiency," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 107–115.

[37] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[38] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[39] H.-C. Wu, "The karush–kuhn–tucker optimality conditions in an optimization problem with interval-valued objective function," *European Journal of Operational Research*, vol. 176, no. 1, pp. 46–59, 2007.

[40] N. Makimoto, I. Nakagawa, and A. Tamura, "An efficient algorithm for finding the minimum norm point in the convex hull of a finite point set in the plane," *OPER RES LETT*, vol. 16, no. 1, pp. 33–40, 1994.

[41] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *International Conference on Machine Learning*. PMLR, 2013, pp. 427–435.

[42] X. Yu, X. Zhang, Y. Cao, and M. Xia, "Vaegan: A collaborative filtering framework based on adversarial variational autoencoders." in *IJCAI*, 2019, pp. 4206–4212.

[43] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, "Modeling user exposure in recommendation," in *Proceedings of the 25th international conference on World Wide Web*, 2016, pp. 951–961.

[44] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[45] A. Gruson, P. Chandar, C. Charbuillet, J. McInerney, S. Hansen, and D. Tardieu, "Offline evaluation to make decisions about playlistrecommendation algorithms," in *WSDM*, 2019, pp. 420–428.

[46] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 42–46.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.