A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation

Yin Zhang*
zhan13679@tamu.edu
Texas A&M University

Derek Zhiyuan Cheng
zcheng@google.com
Google Inc.

Tiansheng Yao
tyao@google.com
Google Inc.

Xinyang Yi
xinyang@google.com
Google Inc.

Lichan Hong
lichan@google.com
Google Inc.

Ed H. Chi
edchi@google.com
Google Inc.

ABSTRACT

Highly skewed long-tail item distribution is very common in recommendation systems. It significantly hurts model performance on tail items. To improve tail-item recommendation, we conduct research to transfer knowledge from head items to tail items, leveraging the rich user feedback in head items and the semantic connections between head and tail items. Specifically, we propose a novel dual transfer learning framework that jointly learns the knowledge transfer from both model-level and item-level: 1. The model-level knowledge transfer builds a generic meta-mapping of model parameters from few-shot to many-shot model. It captures the implicit data augmentation on the model-level to improve the representation learning of tail items. 2. The item-level transfer connects head and tail items through item-level features, to ensure a smooth transfer of meta-mapping from head items to tail items. The two types of transfers are incorporated to ensure the learned knowledge from head items can be well applied for tail item representation learning in the long-tail distribution settings. Through extensive experiments on two benchmark datasets, results show that our proposed dual transfer learning framework significantly outperforms other state-of-the-art methods for tail item recommendation in hit ratio and NDCG. It is also very encouraging that our framework further improves head items and overall performance on top of the gains on tail items.

ACM Reference Format:
Yin Zhang*, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H. Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450086

1 INTRODUCTION

Recommendation systems help users discover things they might be interested in, and have been widely adopted in various online systems, including e-commerce platforms (e.g., [17], [28]), video-sharing and video streaming websites ([52], [20]), and online social platforms ([2], [35]). Building recommendation systems is challenging for many reasons. One of the most notorious problems is the prevalent long-tail distribution in user-item interactions [29]: for items, a small fraction of popular items receive most of the user feedback, while most items only have few user feedback. As an example is illustrated in Figure 1(a), in Netflix, the movie of "Star

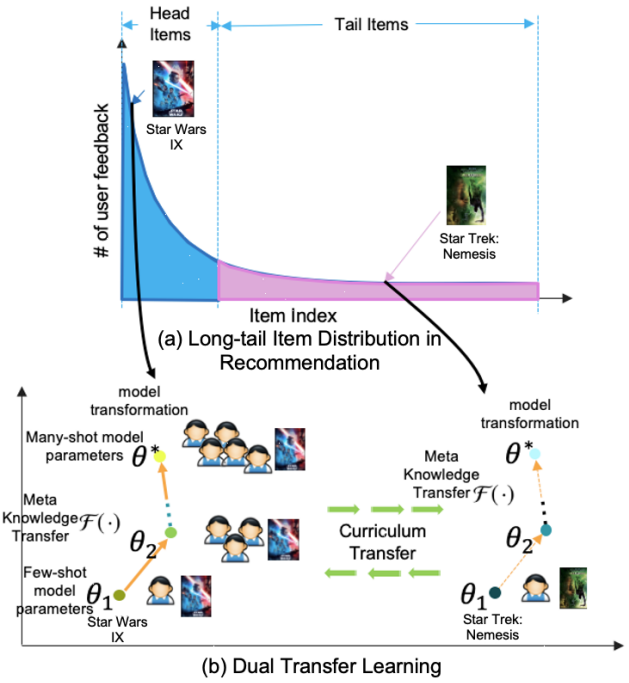


Figure 1: (a) An example of a highly-skewed long-tail distribution for items.
(b) The dual transfer learning framework: one is the meta-level knowledge transfer that learns model parameter changes from few-shot model (recommender trained based on few user feedback) to many-shot model (recommender trained based on rich user feedback), as shown in the orange arrow from θ_1 to θ^*; the other is curriculum transfer, shown in green arrow, that learns feature connections between head and tail items, such as the semantic relations between movie 'Star Wars IX' and movie 'Star Trek: Nemesis'.

*Work done while interning at Google.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450086

Wars IX" contains many user feedback while users give less feedback for movie "Star Trek: Nemesis". As a result, recommendation models trained on dataset with such long-tail item distribution would easily overfit on a small fraction of popular items, and produce sub-optimal performance for the rest of the large chunk of tail items. Deploying such models in real world applications would also lead to the popularity bias [1] and "rich gets richer" feedback loop [31]. Therefore, it becomes critical to address the long-tail item distribution problem in recommendations, especially to improve the recommendation of tail items.

A number of recent studies have aimed to address this problem, including resampling [22] to rebalance the dataset, and reweighting [10] to assign higher costs to tail items in the loss function. Besides these works, in recommendation, the tail items are usually treated as the cold-start items [1]. Those recommenders usually leverage various content information (e.g. item title and cross-domain information) to enhance the representation learning of tail items. However, most existing recommenders focus on the tail items alone, without considering the connection with head items (popular items) – which contain rich user feedback information and transferable contextual information related to tail items (e.g. belonging in the same genres).

In this work, we propose to transfer knowledge from head items that contain rich user feedback information to tail items that have few user feedback, to enhance the recommendation for tail items. Our evaluation criteria is that we expect to improve tail item recommendations while not hurting overall performance. Transferring the knowledge from head to tail items is challenging and non-trivial. Specifically, it poses **two key challenges**:

- First, many existing transfer learning methods are pre-trained on a source task, and then fine-tuned on the final target task. The underlying assumption is that the similar data distribution between source task and target task ensures the high quality of knowledge transfer. This assumption does not suit well for long-tail setting where there is a large gap between the head and tail item distributions [49]. Hence, an important question is how to distill useful knowledge, that can be well transferred to tail items in the context of the imbalanced distribution.
- Second, recommenders usually deal with a large catalog of various items. Head and tail items could show significant diversity in both user preference and content features. Thus another key challenge is how to avoid compromising overall model performance while improving on tail slices.

With these challenges in mind, we propose a novel dual transfer learning recommendation MIRec that contains both (1) Model-level transfer (transfer learning across models) and (2) Item-level transfer (transfer learning across items), as shown in Figure 1 (b). For (1), it learns a generic meta-level knowledge (meta-mapping) on the model-level, i.e., how the model parameters would change when more user feedback is added, highlighted in the orange arrows in Figure 1. As indicated by [49], the learned meta-mapping can capture the implicit data augmentation in model-level without changing data distribution – for example, given an item with a user feedback, the meta-mapping learns to implicitly add similar user who could give feedback to the item. Therefore, the learned meta-mapping can be leveraged to enhance the representation learning of tail items that contains few user feedback. For (2), it considers the divergence between head and tail items. We then utilize the curriculum transfer learning to connect head and tail items. The item-level transfer ensures the learned meta-mapping from head items can be suitable applied to tail items.

The **contributions** of this work are three-fold:

- We propose a novel dual transfer learning framework that collaboratively learns the knowledge transfer from both model-level and item-level. Both components ensure the knowledge can be well utilized from head to tail items in the long-tail distribution settings.
- We design a new curriculum that is capable to smoothly transfer the meta-mapping learned from head items to the long-tail items.
- Extensive experimental results on two benchmark datasets show that MIRec consistently improves tail item recommendations significantly, while achieving solid improvements for the overall performance and head item performance for both hit ratio (HR@K) and NDCG@K. We also find the learned representation of tail items by MIRec preserve better semantics where similar items are close to each other in the embedding space.

2 RELATED WORK

Long-tail Distribution. The long-tail distributions are presented in many real-world datasets, which heavily influences task performance in different areas, such as image classification [10], natural language processing [12], and recommendation [29]. A commonly used strategy for the long-tail distribution problem is resampling to rebalance the dataset, including over-sampling and under-sampling [8, 9, 21, 22]. The over-sampling that adds redundant samples from minor classes easily causes overfit to minor classes, whereas the under-sampling that eliminates the examples from the majority class could lose important samples, leading to sub-optimal performance. Another way to deal with the long-tail distribution is to refine the loss function, such as adding different weights/regularisations for classes/items [6, 10]. For example, the recent proposed logQ corrections [5, 33, 52] construct an item frequency-aware regularizor to alleviate the imbalanced influence. It is also worth to mention that there are some works explore other ways of dealing with the data long-tail distribution, such as meta-learning [49], decoupling the learning process to only adjust classifier [24].

In recommendations, the long-tail distribution of user feedback is more obvious, especially in item-side due to the rapid increase of large amount items. It heavily influences the recommendation performance [6, 13, 29, 54], especially for tail items. Some studies consider the long-tail item distribution to improve the tail item recommendation, such as adding different weights to user-item pairs [54] and clustering [36]. Another close related works are cold-start recommendations [45] that focus on the tail items, with less emphasize on the long-tail distribution. They usually integrate different kinds of user and item side information [27, 45, 55], as well as incorporate different learning methods, such as meta-learning

[1] In the work, cold-start items and tail items are both referred to the same group of items that contain few user feedback.

[14, 26] and active learning [55], to enhance the representation learning.

Different from them, in this work, we focus on transferring knowledge from head items that contain rich user feedback information to help learn tail items in the long-tail distribution, rather than purely based on cold-start items.

Meta-learning in Recommendation. The meta-learning, known as the learning to learn, has attracted many attention due to its powerful performance in many applications. It aims to learn a model that captures the knowledge which can easily adapt to new task. Generally, there are three types of meta-learning approaches: 1. metric-based; 2. model-based; and 3. optimization based. Metric-based approaches [40] learn the distance across different tasks to uncover the general prototype. Model-based approaches [39] focus on the model design to fast learn with few samples. Optimization-based approaches, like the model-agnostic meta-learning (MAML) [16], learn the general knowledge through the optimization among different tasks. Due to the meta-learning success for few-shot learning, recently, there are some attempts that explore the meta-learning framework for cold-start recommendation, such as utilizing the MAML to learn user different optimal parameters [26], incorporating heterogeneous information network by item content information [30], and memory-based mechanism [14].

However, most of those methods focus on utilizing optimization to learn a generalized model (e.g. based on MAML) for cold-start recommendation, which does not fully utilize the long-tail distribution patterns (e.g. with less consideration about the distribution gap between head and tail items in the long-tail item distribution). In this work, we provide another meta-learning method that learns the meta-mapping from few-shot model (trained on small dataset) to many-shot model (trained on large dataset), to transfer the knowledge from head items to tail items. Perhaps the closest work to ours is Wang *et al.* [49] that investigated the image classification with long-tailed distribution dataset. It contains two key learners: base-learner and meta-learner. The base-learner is used to learn the image classification task, where the input is the sample features and output is the class label. The meta-learner learns a meta-level network that capture the model parameter changes from the few-shot model to many-shot model. That is, the meta-learner uses the few-shot model parameters as the input, to map/regress the parameters from many-shot model. Different from it, we adopt the meta-learning in recommendation, where there are substantially large amount of tail items and we further enhanced the meta-mapping to tail items by learning connections among items via curriculum learning.

Curriculum Learning. The curriculum learning, proposed by [4], has shown good performance in different areas [15, 19]. The basic idea is to simulate the human learning process that humans can learn better if the examples are well organized (e.g. knowledge from easy to hard), and apply it to machine learning process. One important question is how to organize samples in the training process rather ran randomly sample, which is known as the curriculum. Recent studies have provided several curriculums and gained a good success, such as the teacher-student mechanics [32], the dynamic curriculum learning [47]. [50] also theoretically showed the curriculum learning can give a robust performance under different conditions. However, few work investigates the curriculum learning in recommendation, which we find especially useful when the long-tail distribution is considered to effectively transfer knowledge among head and tail items.

Table 1: Notation.

Notation	Explanation
$\mathcal{U}$, $\mathcal{I}$	user set, item set
$\mathbf{x}_u$, $\mathbf{y}_i$	user u (item i) feature vector
$I_h(k)$	head item set of items that have no less than k samples
$I_t(k)$	tail item set of items that have less than k samples
Ω^*	training dataset for many-shot model
$\Omega(k)$	training dataset for few-shot model
$g(\cdot)$	base-learner
$\mathcal{F}(\cdot)$	meta-learner

3 DUAL TRANSFER LEARNING FRAMEWORK: MIREC

Problem Statement. Suppose we have a set of users $\mathcal{U}$ and items $\mathcal{I}$, each user $u \in \mathcal{U}$ is represented by the feature vector $\mathbf{x}_u$ with the user ID and various user demographic information. Similarly for items, each $i \in \mathcal{I}$ is represented by the feature vector $\mathbf{y}_i$ with item ID and item content information. Furthermore, frequency of items from $\mathcal{I}$ follow a long-tail distribution, that a small portion of items in $\mathcal{I}$ received most of the feedback from users $\mathcal{U}$, as an example shown in Figure 1 (a). Our goal is to improve recommendation quality on tail slices of the items, while keeping overall performance flat or better. We refer the task as *long-tail recommendation* in the paper. Key notations are summarized in Table 1.

Approach. Considering the long-tail item distribution, inspired by [49], *we propose to transfer knowledge from items that contain rich user feedback (popular items) to items that have few user feedback (tail items)*, for the enhanced long-tail recommendation. Specifically, we propose a dual transfer learning framework that utilizes the long-tail distribution patterns to exploit both meta-knowledge in model level and feature connections in item level for enhanced transfer learning from head to tail:

- Transfer learning *across models*: it learns a meta-mapping from a few-shot model's (model trained on a small datasets [38]) parameters to a many-shot model's (model trained on a large dataset [18]) parameters via a meta-learner $\mathcal{F}(\cdot)$. It captures a generic model meta-level knowledge of model transformation – how model parameters are likely to change with more training data added, to help the learning of tail item that only have few user feedback [18, 49], as shown in Figure 1(b) orange arrows;
- Transfer learning *across items*: it connects head and tail items by their features to smoothly transfer the learned meta-level knowledge between head and tail items. To do so, we design a curriculum that re-organizes the training samples in different curriculum stages to transfer the learned features among head and tail items. The feature connection is used to ensure the learned meta-level knowledge can be applied for the tail items, as shown in Figure 1(b) green arrows.

3.1 Transfer Learning Across Models: Meta-level Knowledge Transfer

The recommender performance usually suffers for tail items. As indicated by [11, 54], one of the main reason is lack of data points [54], and thus cause a weak item representation (tail items are underrepresented). On the other hand, we have fairly rich information and sufficient data points for items on the popular side. Inspired by meta-learning [49], we propose to explore the connection (i.e. meta-learner) between models learned with only a few training examples, and the models learned from the same item but with sufficient training examples through model parameters. Similar as [49], the assumption is that the meta-learner can capture the implicit *data augmentation* [49]. For example, given an item with a user feedback, the meta-learner learns to implicitly add similar users who could have provided feedback to the same item. But rather than explicitly adding similar data point, the meta-learner learns the impact on the learned model parameters – refer as meta-level knowledge. It essentially provides a "magic wand" for us to improve representation quality even when we do not have sufficient data.

In this section, we introduce the meta-level knowledge transfer in our framework, as illustrated in the bottom part in Figure 2. There are two key learners in the meta-level knowledge transfer: (i) **base-learner** $g(\mathbf{x}_u, \mathbf{y}_i; \theta)$: It learns the user preference towards items. It takes a pair of feature vectors $\mathbf{x}_u$ and $\mathbf{y}_i$ for user u and item i as the inputs, with model parameters θ. The output is the user u's preference score towards item i. By using different training data, $g(\mathbf{x}_u, \mathbf{y}_i; \theta)$ is applied for both few-shot model learning and many-shot model learning. (ii) **meta-learner** $\mathcal{F}(\theta; w)$: It learns the meta-mapping from few-shot model parameters to many-shot model parameters. Its input is model parameters θ in $g(\mathbf{x}_u, \mathbf{y}_i; \theta)$ which is trained by $\Omega(k)$, with meta-learner model parameter w.

Base-Learner: With the user and item feature vectors $\mathbf{x}_u$ $\mathbf{y}_i$, the user preference estimation by base-learner $g(\cdot)$ is conducted by a two-tower neural network architecture. It can well encode various types of features for users and items, and is scalable to large corpus dataset [31, 52].

Concretely, as an example shown in Figure 2 in light yellow, the two-tower neural network first learns latent representations (embeddings) given user and item features, through multilayer perceptron (MLP) models (referred as the towers). We denote the user tower as $g_u(\mathbf{x}_u; \theta)$ and item tower as $g_i(\mathbf{y}_i; \theta)$. Then, by combining the user and item embedding through inner product, the output is the user preference towards the item $s(\mathbf{x}_u, \mathbf{y}_i; \theta) =< g_u(\mathbf{x}_u; \theta), g_i(\mathbf{y}_i; \theta) >$.

We formalize the recommendation task as a multi-class classification problem. We use softmax as the loss function to learn the probabilistic distribution of user u's preference towards different items for recommendation, as defined below:

$$p(\mathbf{y}_i|\mathbf{x}_u; \theta) = \frac{e^{s(\mathbf{x}_u, \mathbf{y}_i; \theta)}}{\sum_{j \in \mathcal{I}} e^{s(\mathbf{x}_u, \mathbf{y}_j; \theta)}}. \tag{1}$$

Therefore, the loss function for base-learner $g(\cdot)$ can be formulated as:

$$L_g(\theta|\Omega) = -\frac{1}{|\Omega|} \sum_{(u,i,r(u,i)) \in \Omega} r(u,i) log p(\mathbf{y}_i|\mathbf{x}_u; \theta), \tag{2}$$

where the $r(u, i)$ is the reward function and Ω is the training dataset.

For $r(u, i)$, a binary reward can be defined as:

$$r(u,i) = \begin{cases} 1 & \text{if user } u \text{ gives positive feedback, e.g. click, on item } i, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

By adding the rewards in the loss function, it would learn high preference score for items that user engaged with. Other rewards like the continuous rewards of user engagement (e.g., video watch time), can also be easily integrated. In the following, we focus on the binary rewards for illustration.

For training data Ω, it is generated by a set of triplets $(u, i, r(u, i))$ that contains a user u, item i and their rewards $r(u, i)$. For example,

$$\Omega := \{(u, i, r(u, i))\}.$$

We can use different $(u, i, r(u, i))$ to construct the the training dataset. By turning $g(\cdot)$ on those training dataset with Equation (2), we can get few-shot models (models are trained with few data samples, such as down sample user feedback for each items) and many-shot models (models are trained with rich data samples, such as use all the user feedback). We denote the training dataset that is used to train the few-shot (many-shot) model as $\Omega(k)$ (Ω^*). Details of generating $\Omega(k)$ and Ω^* in MIRec are discussed in section 3.2.

Meta-Learner. With the learned few-shot model and many-shot model by base-learner $g(\cdot)$, we introduce the meta-learner $\mathcal{F}(\cdot)$ that maps the two types of model parameters, to capture the meta-level knowledge – the evolution of model parameter changes when more training examples are added. To do so, we first learn the many-shot model parameters θ^*. Then we present the final objective function that simultaneously trains the few-shot model and meta-mapping.

To learn θ^*, we optimize the $g(\cdot)$ by using Ω^* as the training set:

$$\theta^* = \arg\min L_g(\theta|\Omega^*). \tag{4}$$

where $L_g(\cdot)$ is defined in Equation (2).

With θ^* and $\Omega(k)$, then the final objective function is:

$$L(w, \theta|\Omega^*, \Omega(k)) = ||\mathcal{F}(\theta; w) - \theta^*||^2 + \lambda L_g(\theta|\Omega(k)), \tag{5}$$

where $||\cdot||$ is the l2 norm. λ is the regularization parameter balance the two terms. For each term in Equation (5): (i) $||\mathcal{F}(\theta; w) - \theta^*||^2$ minimizes the distance between the predicted many-shot model parameters (by $\mathcal{F}(\theta; w)$) and many-shot model parameter θ^*. It ensures the predicted many-shot model parameters learned by meta-mapping can well fitted for the data samples in Ω^*; (ii) the second term $L_g(\theta|\Omega(k))$ trains the base-learner based on training dataset $\Omega(k)$, to encourage the high recommendation performance for data samplings in $\Omega(k)$. The $L_g(\cdot)$ is the same as Equation (2). A good performance of base-learner could help the learning of meta-mapping that also maintained high accuracy on the base recommendation task. Therefore, the loss function that simultaneously trains the few-shot model and meta-mapping offers both the high-quality of base-learner recommendation and high accuracy of meta-learner model mapping.

Implementations of $\mathcal{F}(\cdot)$. In practice, the base-learner $g(\cdot)$ could be very complex, such as the two-tower model where each tower could contain many layers. In this situation, $g(\cdot)$ has a large number of model parameters. Directly mapping all those $g(\cdot)$ model

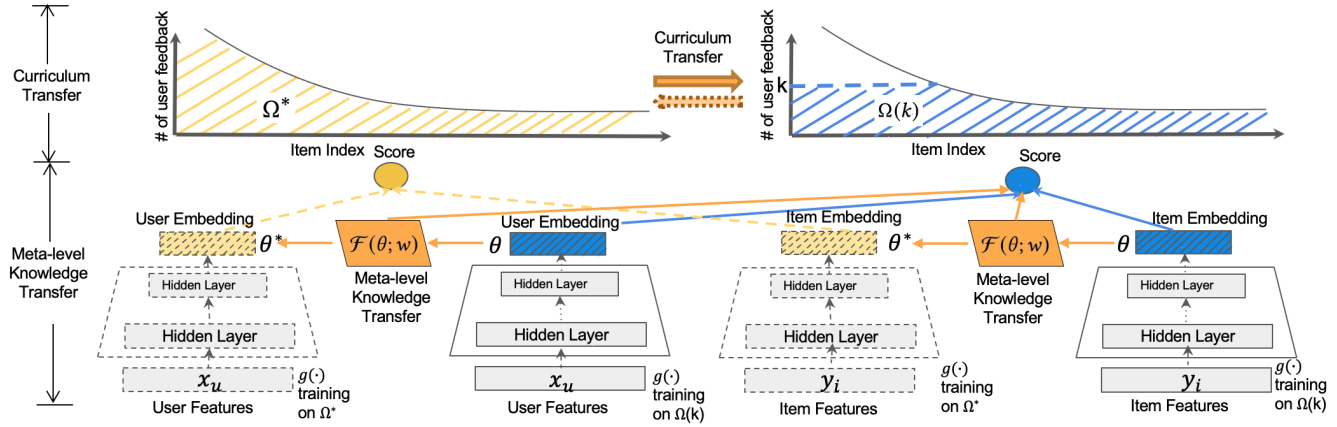


Figure 2: Framework of MIRec. It contains two types of transfers: (i) via the meta-learner $\mathcal{F}(\theta; w)$, the meta-level knowledge transfer learns the model parameter evolution from few-shot model parameters θ to many-shot model parameters θ^*; (2) the curriculum transfer learns the connections between head and tail items to smooth transfer the meta-level knowledge from head to tail.

parameters by $\mathcal{F}(\cdot)$ could be both high computational cost and unstable [3, 48]. Therefore, similar as [49], we focus on the model parameters in the last layer of user and item towers for meta-learning. In this way, the meta-learning is capable to be more efficient and generalized to learn different items, which is especially helpful for the large number of various tail items in recommendation. For $\mathcal{F}(\cdot)$, different networks can be applied, such as regression networks [48], knowledge graph [37], and DNN-based network [42]. Here we use a fully connected layer as the mapping network for $\mathcal{F}(\cdot)$, which we empirically find is a relative simple and effective approach that can gain good performance.

3.2 Transfer Learning Across Items: Curriculum Transfer

As shown in Section 3.1, both the training dataset $\Omega(k)$ for few-shot model and Ω^* for many-shot model play a pivotal role to ensure the high quality of learned model generic transformation by $\mathcal{F}(\cdot)$.

A traditional way to construct the $\Omega(k)$ and Ω^* is only based on the head items. For example, [49] selects items that have no less than k user feedback, denoted as $I_h(k)$, and uses all the user feedback from item $i \in I_h(k)$ to construct the training dataset for many-shot model. The few-shot model is learned by randomly sampling a smaller fixed number of user feedback to $i \in I_h(k)$. Therefore, the meta-knowledge of model parameter evolution is only based on the head items. Since the learned meta-mapping is directly applied on the few-shot models of tail items $i \in I_t(k)$ ($I_t(k)$ is the set of items that have less than k user feedback), a basic assumption for the directly mapping is that the meta-mapping for head and tail items are the same.

However, different from many other classification problems (e.g. image classification), the number of items (i.e., classes) for recommenders is highly diverse. In addition, there are way more tail items with few observations than the head items. In this situation, the meta-mapping that are learned only based on the head items would constrain the mapping of the large amount of tail item representations, leading to sub-optimal performance. Thus, we introduce curriculum learning to smoothly transfer the meta-mapping from head items to tail items by construction Ω^* and $\Omega(k)$.

Concretely, we propose a Long-Tail distribution-aware CURriculum strategy (LTCur) that can both well transfer the learned features from head items to tail items, and at the same time, relatively keeping the original distribution to alleviate the influence of distribution bias. Therefore, we construct the two sets Ω^* and $\Omega(k)$ to ensure both head and tail items are trained in the training stages.

In LTCur, the training set Ω^* for many-shot model is defined as:

$$\Omega^* := \{(u, i, r(u, i))\} \tag{6}$$

that $\forall(u, i, r(u, i)) \in \Omega^*$ satisfies:

- $u \in \mathcal{U}, i \in \mathcal{I}$;
- $\sum_{(u,i,r(u,i))\in\Omega^*} r(u, i) = \sum_{u\in U} r(u, i)$, for $i \in \mathcal{I}$;

That is, as shown in Figure 2 of item distribution in yellow part, Ω^* includes the all the user feedback from both the head items and tail items. To train $g(\cdot)$ on Ω^*, considering the large number of tail items, we add the logQ correction [52]:

$$s^c(\mathbf{x}_u, \mathbf{y}_i; \theta^*) = s(\mathbf{x}_u, \mathbf{y}_i; \theta^*) + \lambda_c \, log(p_i), \tag{7}$$

where the λ_c is the weight for the correction. p_i is the sampling probability of item i, which is usually calculated by the frequency of item i divided the total number of training samples.

For the training set $\Omega(k)$ that is used to train the few-shot model parameters, as shown in Equation (5), we construct it as:

$$\Omega(k) := \{(u, i, r(u, i))\} \tag{8}$$

that $\forall(u, i, r(u, i)) \in \Omega(k)$ satisfies:

- $u \in \mathcal{U}, i \in \mathcal{I}$;
- $\sum_{(u,i,r(u,i))\in\Omega(k)} r(u, i) = k$, if $i \in I_h(k)$;
- $\sum_{(u,i,r(u,i))\in\Omega(k)} r(u, i) = \sum_{u\in U} r(u, i)$, if $i \in I_t(k)$;

that is, we consider all the user feedback for the tail items. The new constructed training set Ω^* and $\Omega(k)$ ensure (1) tail items are fully trained in both the many-shot model and few-shot model to ensure

Algorithm 1: Training procedure of MIRec.

Input : user set $\mathcal{U}$, item set $\mathcal{I}$, feature vector $\mathbf{x}$, $\mathbf{y}$ that maps each user and item input to an embedding spase, step size α, β, γ, batch size B

Initialized Model parameters θ, w;
Construct sets Ω^*, $\Omega(k)$ based on Equation (6) and (8);
Function OPTIMIZE_PARAMETERS(Ω, $g(\cdot)$):
  # Calculate the optimize model parameters for many-shot model;
  randomly initialize θ^*;
  while *not converged* **do**
    Sample batch of user and item pairs $\{(u, i, r(u, i))\}_{m=1}^{B} \in \Omega$
    for *each* $(u, i, r(u, i))$ **do**
      $\theta^* \leftarrow \theta^* - \alpha \bigtriangledown_{\theta^*} L_g(\theta^*|\Omega)$
  return θ^*;
$\theta^* \leftarrow$ OPTIMIZE_PARAMETERS(Ω^*, $g(\cdot)$);
while *not converged* **do**
  Sample batch of user and item pairs $\{(u, i, r(u, i))\}_{m=1}^{B} \in \Omega(k)$
  for *each* $(u, i, r(u, i))$ **do**
    for *local update steps* **do**
      $\theta \leftarrow \mathcal{F}(\theta; w)$;
      calculate $L_g(\theta|\Omega(k))$;
      $\theta \leftarrow \theta - \beta \bigtriangledown_{\theta} L(w, \theta|\Omega^*, \Omega(k))$
    Global update both meta-learner model parameters w in $\mathcal{F}(\cdot)$ and few-shot model parameter θ based on Equation (5): $w \leftarrow w - \gamma \bigtriangledown_{w} L(w, \theta|\Omega^*, \Omega(k))$;
    $\theta \leftarrow \theta - \gamma \bigtriangledown_{\theta} L(w, \theta|\Omega^*, \Omega(k))$;
return θ, w

the high quality of the learned item representations in both many-shot and few-shot models. It can further act as the cornerstone for the high performance of meta-mapping. (2) In the few-shot model training, the distribution of tail items relatively keeps the same as the original distribution. It can alleviate the bias among tail items that brings by the new distribution.

The two training datasets Ω^* and $\Omega(k)$ both contain the head and tail items but with different head and tail distributions. Thus, based on the curriculum learning [4, 11], the meta-mapping $\mathcal{F}(\cdot)$ not only learns the mapping from few-shot model parameters to many-shot model parameters, but also feature connections between head and tail items through the curriculum. The dual transfer framework ensures the smooth meta-mapping transfer from head to tail items. Experiments verify that our proposed curriculum can improve the performance of both overall and tail item recommendation performance. Details of the implementation of MIRec is shown in Algorithm 1.

3.3 Prediction by MIRec

The meta-level knowledge transfer learns the model parameter evolution when more training data is available. Different from tail items, head items contain rich user feedback. since $\mathcal{F}(\cdot)$ is learned to match θ^* by Equation (5), the base-learner could offer a better representations for head items rather than ones by meta-mapping. Therefore, to predict the user preference towards both head and tail items, we combine the representation learned from both the base-learner and meta-mapped base-learner [49]. One example to integrate their representations for user preference learning is:

$$\begin{aligned} s_{MIRec}(\mathbf{x}_u, \mathbf{y}_i) &= s(\mathbf{x}_u, \lambda_p * \mathbf{y}_i; \theta^*) + s(\mathbf{x}_u, (1-\lambda_p) * \mathbf{y}_i; \mathcal{F}(\theta; w)) \\ &= \lambda_p g(\mathbf{x}_u, \mathbf{y}_i; \theta^*) + (1-\lambda_p) g(\mathbf{x}_u, \mathbf{y}_i; \mathcal{F}(\theta; w)), \end{aligned} \tag{9}$$

where λ_p is used to balance the meta-mapping based model and many-shot based model. And we experimentally find it has a good performance.

4 DISCUSSION

Expansion of MIRec. In this work, we construct two subsets Ω^* and $\Omega(k)$ for transfer learning under the long-tail distribution, where k is a fixed constant. To make the transfer learning more smooth among items, we can further vary the k to construct different sets and recursively learn the model dynamic trajectories. Concretely, we can select a sequence of $\{k_j\}$ that satisfies $k_1 < k_2 < k_3, \ldots < k_m$. Then the corresponding $\Omega(k_j)$ is constructed based on Equation (8). The meta-learner would be a sequence of $\mathcal{F}_{1,2}(\cdot), \ldots \mathcal{F}_{j,j+1}(\cdot) \ldots \mathcal{F}_{m-1,m}(\cdot)$, where $\mathcal{F}_{j,j+1}(\cdot)$ that learned the model mapping from $\Omega(k_j)$ to $\Omega(k_{j+1})$ in a recursive way [49]. And the final prediction of user u preference towards item i can be calculated by:

$$\lambda_{p,0} g(\mathbf{x}_u, \mathbf{y}_i; \theta^*) + \sum_{j=1}^{m-1} \lambda_{p,j} g(\mathbf{x}_u, \mathbf{y}_i; \mathcal{F}_{j,j+1}(\theta_{j,j+1}; w_{j,j+1})),$$
$$where \sum_{j=0}^{m-1} \lambda_{p,j} = 1.$$

Long-tail distribution from user side. So far we focus on the long-tail distribution from item side. In practice, MIRec can also be adopted when we focus on the long-tail distribution from user side or from both user and item side. When only the user side long-tail distribution is considered, we can simply change the MIRec by reversing the user u and item i. And the Ω^* and $\Omega(k)$ is constructed based on the user distribution, correspondingly. When the long-tail distribution is considered from both user and item sides, one simple way that we can do is to use both user- and item-based MIRec, and integrate their results as many ensemble methods.

5 EXPERIMENT

In this section, we evaluate MIRec for recommendation tasks through experiments on two datasets. We seek to address the following key research questions:

RQ1: How well does the dual transfer learning framework MIRec perform compared to the state-of-the-art methods?

RQ2: How do different components (meta-learning and curriculum learning) of MIRec perform individually? Could they complement each other?

Table 2: Summary of the datasets and content information.

	MovieLens1M	Bookcrossing
# User	6,040	50,454
# Items	3,706	222,154
# Feedback	1,000,209	1,031,175
Sparsity	95.5316%	99.9908%
User Features	IDs, Gender, Occupation, ZipCode, Age	IDs, Location, Age
Item Features	IDs, Title, Genres, Year	IDs, Title, Author, Year, Publisher

RQ3: How does our proposed curriculum learning strategy compare with the alternatives?

RQ4: Besides downstream task performance, are we actually learning better representations for tail items? Could we see the differences visually?

5.1 Datasets

We adopt two widely used benchmark datasets: MovieLens 1M [2] and Bookcrossing [3], both with rich user and item information. We follow the similar procedures adopted in [23, 25, 26, 43] to engineer the features and labels. Specifically, all the <user, item> pairs with an explicit ratings are considered as positive examples (1s), and the rest of the pairs are all considered as negative examples (0s). Among these, any pairs of <user, item> with invalid / missing features are filtered, as dealing with the missing value is often treated as a separate issue [34, 53]. Besides age and year which are treated as the continuous features, all the other user and item features are treated as categorical feature and represented by one-hot encoding. We apply bag-of-words representation for item title feature [41]. A summary of the two datasets is shown in Table 2.

In both datasets, the item distribution follows a highly-skewed long-tail distribution (see Figure 3). Especially for the bookcrossing dataset, a small portion of popular items contributes more than half of all the user feedback. It shows the two datasets are well-suited for our target problem of improving tail recommendation quality in the presence of long-tail item distribution.

Consistent with the prior work [23, 25], we divide the data into three parts by leave-one-out evaluation: for each user, the most recent item is for testing, the second most recent interacted item is for validation, and the rest of the items are for training. Since the Bookcrossing dataset does not have the timestamp information, for each user, we randomly select one item for testing, one for validation, and use the rest of items for training.

5.2 Setup

Evaluation Criteria. We adopt two common evaluation metrics, Hit Ratio at top K (HR@K) and NDCG at top K (NDCG@K) [23]. HR@K measures whether the test item is retrieved in the top K ranked items. The NDCG@K considers the position of correctly recommended items by giving higher scores to the top ranked items.

With the goal of improving the tail item recommendation quality without hurting the overall performance, we further report the metrics evaluated on the tail and head item set respectively. Based on the Pareto Principle [7] and the item distributions (Figure 3), we split the first 20% most frequent items in Movielens1M and 0.1% items in Bookcrossing [4] for head items, and the rest items are treated as tail items.

[2]https://grouplens.org/datasets/movielens/1m/

[3]http://www2.informatik.uni-freiburg.de/~cziegler/BX/

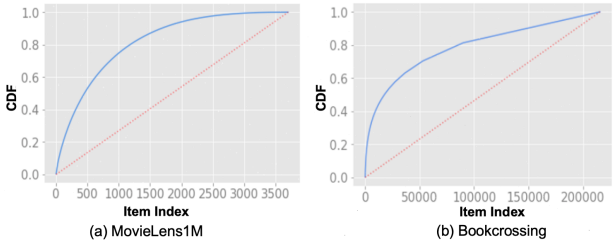


Figure 3: The CDF of the item distribution in MovienLens1M and Bookcorssing. It shows items in both datasets are in the long-tail distributions.

Table 3: Evaluation criteria w.r.t. performance improvement on different data slices.

HR@K/NDCG@K	Overall	Head Items	Tail Items
Good Results	–	↘	↗
Better Results	↗	–	↗
Great Results	↗	↗	↗

The two metrics above are usually used to evaluate the overall model performance, regardless of the data distribution. Our goal of the framework is to build a single model to improve the tail item recommendation without hurting the overall performance by considering the long-tail item distribution. Thus, we report model performance on head, and tail slices respectively, as well as the overall performance. We describe how we claim success for our recommenders in Table 3, with the assumption that we are building one single model to serve both head and tail recommendations. Alternatively, if we were to build separate models for head and tail slices, most likely we would be able to achieve better results from both models on their corresponding data slices.

As shown in the Table 3, we consider a recommender has 'good results' when the model improves on the tail, while taking a hit on the head but still keeping overall performance flat. The results are even 'better' when we could keep the performance on head items neutral while improving both on tail and overall. The best scenario would be achieving improvements across the board on head, tail, and overall.

Baselines. To fully demonstrate the effectiveness of MIRec on long-tail item distribution, we compare MIRec with comparative model-agnostic methods, which includes strategies such as training distribution re-sampling, loss function refinement and various curriculum learning and meta-learning strategies. For fair comparison, similar as [31], we consider the same content-aware two-tower DNN model [52] as the backbone model to establish a reasonable benchmark. The same number of layers and dimensions are used for all the strategies and MIRec for fair comparison. We list all the baseline models here:

Backbone Model:

[4]The split rate of 0.1% can better show the performance differences among methods, especially for tail items.

- *Two-tower Model* [46, 51, 52]: This model architecture has been proven to be highly effective in content-aware settings. It also demonstrates great scalability due to the efficiency of inner product for inference.

Re-sampling Strategies:

- *Over-sampling* [8]: A training data re-sampling strategy that repeatedly samples from tail items to re-balance the head items and tail items in the training distribution. The over-sampling strategy is more common in practice since user feedback data is highly valuable, and we do not want to down-sample on the positives.
- *Under-sampling* [8]: In contrast to over-sampling, we keep the tail items unchanged, and down-sample the head items.

Loss Function Refinement:

- *ClassBalance* [10]: This is a recent state-of-the-art method for tackling the long-tail distribution in image classification settings. It first calculates the effective number of samples for each imbalanced class. Then a class balanced loss function is designed by using different weights of each class based on the effective number of samples. We adopt this approach method to the recommendation task, where items are treated as classes.
- *LogQ* [33]: The logQ correction is widely used to address the long-tail distribution problem in different areas. It corrects the predicted logits in the loss function by adding a correction term based on estimated item frequency.

Curriculum Learning Strategies:

- *Head2Tail*: Based on the previous studies [6], the recommendation performance on tail items is significantly worse than the head items, and the prediction variances of tail items are high. Thus, following the curriculum learning idea, we first train the two-towel model with head items in the first curriculum stage, then fine-tune the model with tail items only in the second curriculum stage.
- *Tail2Head*: We also explore the recommendation performance of Tail2Head which adopts the exact opposite of the curriculum of Head2Tail. That is, it first trains on the tail items, then fine-tunes the model on the head items.

Meta-learning Strategies:

- *MeLu* [26]: This is a recent state-of-the-art meta-learning based method for cold-start item rating prediction. It applied the optimization-based meta-learning (MAML) for rating prediction in the cold-start problem. We modify it to be applicable for the implicit feedback prediction task. For a fair comparison, we also applied the two-tower model as the backbone model for MeLu.

Hyper-parameter settings. We set the embedding dimension as 64 across all the different baselines and treatment models. For approaches that adopted the two-tower model architecture, we used exactly the same numbers of hidden layers, and hidden units. Concretely, we employ a widely used tower structure, where hidden dimensions of higher layers have 1/2 number of neurons of its next lower layer. The ReLU function is used as the active function. For methods that consider curriculum learning, we consider 100 training epochs for each curriculum stage (two stages in total) for a fair comparison. For non-curriculum-based methods, the total number of epochs is 200. The decay rate between each curriculum stages is determined by grid search from {0.1, 0.01, 0.001, 0.0001}. Similar as the other meta-learning [26], we perform one gradient decent for local update in each step. The k in $\Omega(k)$ is determined by the split of head and item items. Regularization parameters are determined by grid search in the range of {0.3, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001}. Dropout rate and learning rate are determined by grid search in the range of {0.1, 0.3, 0.5, 0.7, 0.9} and {0.1, 0.01, 0.001, 0.0001, 0.00001} respectively. The batch size is set to 1024, to be consistent with previous results [52]. The Adaptive moment estimation (Adam) is used to optimize MIRec. MIRec is implemented with TensorFlow Recommenders (TFRS) [5].

5.3 Recommendation Performance (RQ1)

We first compare methods for top-K recommendation on head, tail items and overall performance. Then we vary the key hyper-parameter – embedding dimension – to further investigate the MIRec performance on recommendation.

5.3.1 TopK Performance. The top-K recommendation performance for all items, head and tail items are shown in Table 4 for MovieLens1M and Table 5 for Bookcrossing. The user and item embedding dimensions for all methods are set to be the same for fair comparison. Overall, we see that MIRec generally brings the improvement in HR@K and NDCG@K for both tail and head items, and the overall performance. This is very encouraging since it's very hard to improve both tail and head items at the same time. Concretely, we have the following **key observations**:

- First, compared with the state-of-the-art two-tower model, the MIRec generally achieves great improvements for both tail items and head items. Since both methods are based on the two-tower architecture, the results verify the importance of considering the long-tail item distribution in recommendation prediction. The improvements of both head and tail items also confirm that the dual transfer learning framework indeed benefits the learning of items in the long-tail distribution for enhanced recommendation.
- Second, among different strategies for long-tail distribution problem (e.g. re-sampling, re-weighting, and transfer learning), MIRec achieves the best performance. The results demonstrate the superiority of considering transfer learning among items in the long-tail item distribution. Concretely, as shown in Table 4 and Table 5, the re-sampling strategies (i.e. over-sampling and under-sampling) show poor performance in recommendation. The results are consistent with the previous findings [49, 52] that re-sampling could heavily influence the model performance and cause sub-optimal performance due to the changes of the original data distribution. It also illustrates the importance of relatively keeping the original data distribution as it was in the training process. For the refining loss function strategies (e.g. logQ), they could achieve good performance for tail items, but perform quite neutral

[5] https://blog.tensorflow.org/2020/09/introducing-tensorflow-recommenders.html

Table 4: The recommendation performance of MIRec versus baselines on MovieLens1M.

Measure%	Overall		Head		Tail	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Two-tower	6.74	3.26	8.98	4.33	3.22	1.59
Over-sampling	0.23	0.10	0.14	0.04	0.37	0.19
Under-sampling	0.72	0.33	0.41	0.20	1.19	0.53
ClassBalance	6.32	2.98	9.10	4.43	1.45	0.71
LogQ	2.24	0.95	1.41	0.54	3.53	1.59
Head2Tail	1.31	0.66	1.90	0.97	0.40	0.18
Tail2Head	1.71	0.85	2.74	1.34	0.09	0.07
MeLu	5.96	2.88	8.98	4.35	1.23	0.58
MIRec	**7.02**	**3.36**	**9.13**	**4.36**	**3.70**	**1.81**

Table 5: The recommendation performance of MIRec versus baselines on Bookcrossing.

Measure%	Overall		Head		Tail	
	HR@100	NDCG@100	HR@100	NDCG@100	HR@100	NDCG@100
Two-tower	4.98	1.56	54.67	17.58	0.32	0.05
Over-sampling	0.17	0.03	0.51	0.12	0.14	0.02
Under-sampling	0.76	0.16	3.56	0.73	0.51	0.11
ClassBalance	4.58	1.06	49.62	11.77	0.45	0.08
LogQ	2.06	0.49	15.52	4.11	0.82	0.15
Head2Tail	2.66	0.58	24.68	5.66	0.65	0.06
Tail2Head	4.82	1.07	52.00	12.73	0.03	0.01
MeLu	3.33	0.75	30.28	7.22	0.86	0.16
MIRec	**5.25**	**1.60**	**55.28**	**17.29**	**0.58**	**0.14**

or poorly for head items. It is reasonable since logQ can give more weights to the tail items in the loss function. The results demonstrate the clear advantage of our dual transfer learning framework. In Table 5, for baselines that have better performance on tail items (e.g. LogQ and Head2Tail), their performance significantly decreases for head and overall performance. A possible reason is their learned feature representations are sub-optimal, causing an unhealthy trade-off between tail items and head items. Different from them, MIRec is able to achieve significant gains for tail items while maintaining or sometimes improving the head/overall item performance. This further shows the high-quality of learned representations by MIRec.

- Third, compared with the commonly used curriculum learning strategies (Head2Tail, Tail2Head) and state-of-the-art meta-learning (MeLu), MIRec obtains better performance. It shows by integrating curriculum learning with meta-learning, the dual transfer framework can better transfer knowledge in the long-tail item distribution. Note here the MeLu does not perform good, one possible reason is that it does not well fitted for the implicit feedback based recommendation. MeLu also requires a large amount memory when high dimensional item features are considered.

5.3.2 Influence of Embedding Dimension. We also analyze the key hyper-parameter – the embedding dimension – on MIRec to see its effect on MIRec's performance. Results for the MovieLens1M dataset is shown in Figure 4, with comparison of other representative methods. As shown in Figure 4, MIRec consistently outperforms the other methods when different embedding dimensions are considered. It further confirms the importance of considering long-tail item distribution and utilizing the transfer learning to learn the item relations in the long-tail distribution for improved recommendation. It is also worth noting that performance of tail items tends to degrade with higher embedding dimension. Our hypothesis is that tail items are easier to get overfitted due to having less data for training.

5.4 Dual Transfer Learning (RQ2)

Given the great performance of MIRec, we further investigate how does each component in MIRec contribute to the improved performance on long-tail item recommendation. Specifically, we conduct an ablation study experimenting different versions of MIRec that only includes a specific component:

- $MIRec_M$: This only considers the meta-learning part in the MIRec. Similar as [49], Ω^* and $\Omega(k)$ are constructed only with head items $i \in I_h(k)$. Concretely, Ω^* is built by all the user feedback of item $i \in I_h(k)$. We randomly sampling k user feedback of each $i \in I_h(k)$ to construct $\Omega(k)$.
- $MIRec_C$: This only considers the curriculum learning part in MIRec. Specifically, the two-tower model is first trained based on Ω^*, then fine-tuned on $\Omega(k)$.
- $MIRec - LogQ$: We use the $s(x_u, y_i)$ to calculate the user preference towards items instead of the logQ correction based $s^c(x_u, y_i)$ in Equation (7);
- $Two - tower_2$: MIRec uses both the many-shot model and meta-mapped model to predict the user preference (as shown in

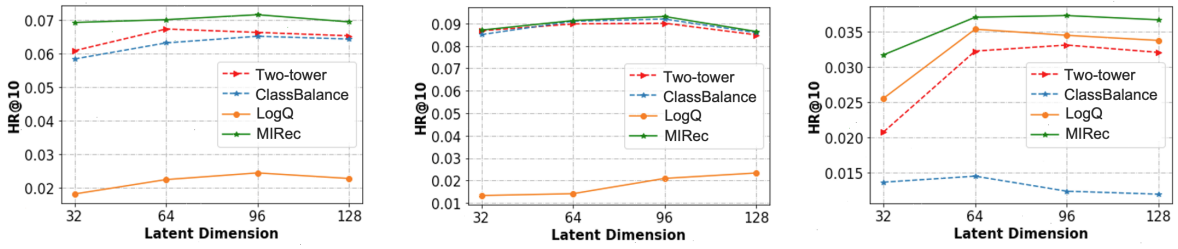


Figure 4: Recommendation performance (overall, head and tail) for different latent dimensions on MovieLens1M dataset.

Equation (9)). Therefore, we use two two-tower models that have similar number of model parameters as MIRec for further comparison. That is, the prediction is based on the equation $s'(\mathbf{x}_u, \mathbf{y}_i) = \lambda_p g(\mathbf{x}_u, \mathbf{y}_i; \theta^*) + (1 - \lambda_p) g(\mathbf{x}_u, \mathbf{y}_i; \theta')$ where θ' is calculated by another two-tower model with different initialization.

The results of the ablation study is shown in Table 6, that MIRec outperforms the other variations. This is to be expected since we expect all different components contribute to the improved performance.

Comparing $MIRec_C$ and $MIRec_M$, which only use one type of transfer learning, we see that $MIRec_M$ that uses the vanilla meta-learning method performs poorly. This indicates that directly applying the meta-mapping, which is learned on the head items only, could limit the learning of various tail item's representation. It also highlights the importance of considering curriculum transfer among items to make the learned meta-mapping suitable for tail items.

$MIRec_C$ achieves good performance for tail items, demonstrating the proposed curriculum (from Ω^* to $\Omega(k)$) indeed transfers knowledge from head to tail items to improve the performance of tail items. We also observe that $MIRec_C$ does not perform well on head items, which is expected. One hypothesis is that since we down-sample head items in the second phase, the model might have forgotten some of the patterns learned on head items.

Furthermore, comparing $MIRec{-}LogQ$ with MIRec, MIRec achieves a better performance, that verifies that logQ correction can further help the knowledge transfer in the long-tail item distribution by adding the item frequency based correction.

Lastly, for $Two-tower_2$, though it contains similar number of learned parameters as MIRec, $Two-tower_2$ performs poorly for tail items, compared with MIRec. One likely reason is that the combined two two-towel models in $Two-tower_2$ easily causes over-fitting for tail items. It further shows the importance of transfer knowledge in long-tail item distribution to improve the recommendation for tail items.

5.5 Curriculum Learning Process (RQ3)

The proposed curriculum learning – Long-Tail distribution-aware CURriculum (LTCur) plays a pivotal role in transferring knowledge among head and tail items in MIRec. Thus, in this section, we further study the performance of curriculum learning strategies on both head and tail items in different curriculum stages. Specifically, we report the loss values on head & tail slices, and overall performance over time.

In Figure 5, each row represents a curriculum strategy. x-axis represents the epoch index, and y-axis reports loss values. Each curriculum contains two curriculum stages. Each line is *smoothed*. The training loss values (red line) is 0 if the head/tail items are not trained. For example, in Head2Tail, since the tail items in the training dataset are not trained, the training loss of tail items (row 1 column 3 in the red line) is 0 in the first curriculum stage, similar for the head items in the second stage (row 1 column 2 in the red line).

From the Figure 5, we observe that: (1) Compared to the tail item loss in different curriculums (column 3), our proposed curriculum can bring a two-stage decent for both the training and validation loss, as shown in row 3 column 3 in Figure 5. It demonstrates the LTCur curriculum enhances the learning of tail items in both two curriculum stages. (2) When the model is trained based on only head/tail items, the validation performance for the other part of items decreases. The different changes of head and tail loss indicate the large variations between head and tail items, which further confirms the importance of transferring knowledge between head and tail items for enhanced recommendation. (3) It is easily to get validation loss increases if the model is trained purely based on head/tail items. As shown in first column of the first two rows: when the training loss decreases (red line), the validation loss would increase (blue line). It highlights the importance of considering both head and tail items in different curriculum stages.

5.6 Embedding Visualization and Case Study (RQ4)

In this section, we look into the learned representations from MIRec by visualizing the embeddings, and examine the performance of MIRec qualitatively.

First, we visualize the learned tail item embeddings from the baseline (two-tower model) and MIRec using t-SNE (Figure 6) and highlight movie genres by different colors. We observe that the movie clusters from MIRec, compared with the baseline model, is more coherent w.r.t. the movie genres information. This suggests that the improved performance from MIRec can be attributed to better capturing the semantic information of tail items.

Next, we further look into what movies that MIRec clusters together and visualize the clusters by movie titles in Figure 7. Here we observe similar movies are clustered together. For example, 'Billy's Holiday' and 'Juno and Paycock', which are both about people's life, are clustered together. The visualization shows that although the tail items contain less collaborative information, MIRec can still well cluster those tail items based on their content information. This further validates our previous findings on MIRec.

Table 6: Ablation study of MIRec on MovieLens1M.

Measure%	Overall		Head		Tail	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Default	6.74	3.26	8.98	4.33	3.22	1.59
MIRec	**7.02**	**3.36**	**9.13**	**4.36**	**3.70**	**1.81**
$MIRec_M$	6.48	3.13	8.75	4.17	2.88	1.43
$MIRec_C$	6.83	3.30	8.77	4.21	3.78	1.87
$MIRec - LogQ$	6.98	3.36	9.75	4.71	2.62	1.26
$Two - tower_2$	6.90	3.22	9.46	4.37	2.89	1.41

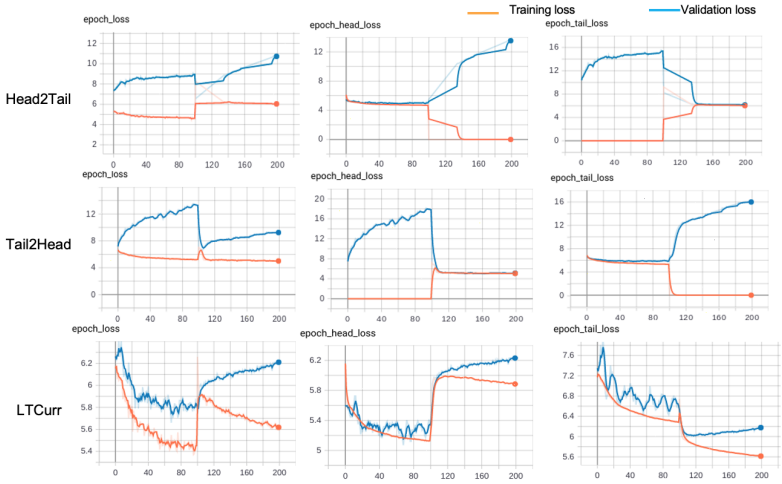


Figure 5: The loss changes of different curriculums (by rows) for overall items, head items and tail items (by columns) on MovieLens1M. x-axis is the index of epoch. Blue/red line shows the loss of validation/training dataset.

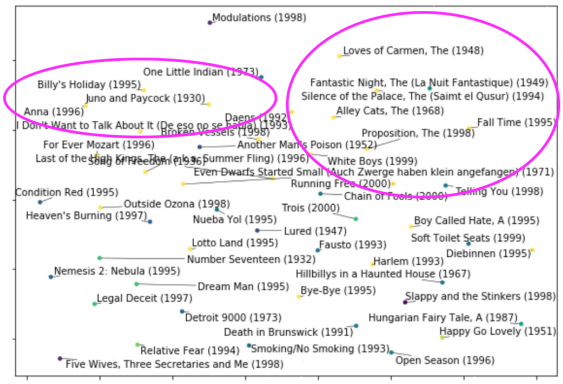
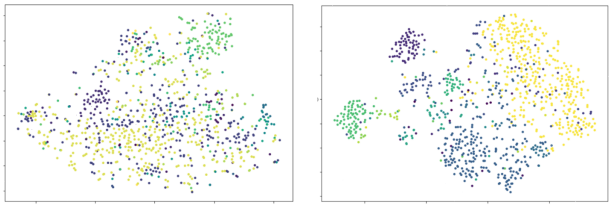


Figure 7: Case study: movies are clustered together by MIRec. The two purple circles highlight movies with similar themes are close to each other in the embedding space learnt by MIRec.



(a) Two-tower Model (b) MIRec

Figure 6: The 2-D visualization (with t-SNE [44]) of tail items on MovieLens1M. The color represents the movie genres.

6 CONCLUSION

In this work, we tackle the challenge of recommendations in the context of long-tail item distribution. We propose a dual transfer learning framework – MIRec – that integrates both (i) meta-learning to transfer knowledge among models, and (ii) curriculum learning to transfer knowledge among items. The proposed curriculum learning inside the meta-learning ensures the smooth knowledge transfer from head items to the large number of tail items in recommendation. Extensive experiments on the two benchmark datasets show that MIRec consistently outperforms the state-of-the-art algorithms. For future work, we are interested in (i) improving the curriculum learning inside the meta-learning approach, and (ii) exploring other types of content information, such as incorporating item knowledge graph with the long-tail item distribution.

ACKNOWLEDGMENTS

The authors would like to thank Maciej Kula, Jiaxi Tang, Wang-Cheng Kang for their help in this work.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*.

[2] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. 2011. Analyzing user modeling on twitter for personalized news recommendations. In *international conference on user modeling, adaptation, and personalization*. Springer.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*.

[5] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008).

[6] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond globally optimal: Focused learning for improved recommendations. In *Proceedings of the 26th International Conference on World Wide Web*.

[7] George EP Box and R Daniel Meyer. 1986. An analysis for unreplicated fractional factorials. *Technometrics* 28, 1 (1986).

[8] Jason Brownlee. 2020. *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002).

[10] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[11] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. 2018. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[12] Paula Czarnowska, Sebastian Ruder, Edouard Grave, Ryan Cotterell, and Ann Copestake. 2019. Don't Forget the Long Tail! A Comprehensive Analysis of Morphological Generalization in Bilingual Lexicon Induction. In *EMNLP*.

[13] Marcos Aurélio Domingues, Fabien Gouyon, Alípio Mário Jorge, José Paulo Leal, João Vinagre, Luís Lemos, and Mohamed Sordo. 2013. Combining usage and content in an online recommendation system for music in the long tail. *International Journal of Multimedia Information Retrieval* 2, 1 (2013).

[14] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[15] Rasheed El-Bouri, David Eyre, Peter Watkinson, Tingting Zhu, and David Clifton. 2020. Student-teacher curriculum learning via reinforcement learning: predicting hospital inpatient admission location. In *ICML*.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

[17] Asnat Greenstein-Messica and Lior Rokach. 2018. Personal price aware multi-seller recommender system: Evidence from eBay. *Knowledge-Based Systems* 150 (2018).

[18] Liang-Yan Gui, Yu-Xiong Wang, Deva Ramanan, and José MF Moura. 2018. Few-shot human motion prediction via meta-learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[19] Yong Guo, Yaofo Chen, Yin Zheng, Peilin Zhao, Jian Chen, Junzhou Huang, and Mingkui Tan. 2020. Breaking the curse of space explosion: Towards efficient NAS with curriculum search. In *ICML*.

[20] Blake Hallinan and Ted Striphas. 2016. Recommended for you: The Netflix Prize and the production of algorithmic culture. *New media & society* 18, 1 (2016).

[21] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE.

[22] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009).

[23] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*.

[24] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *ICLR*.

[25] Wang-Cheng Kang, Derek Zhiyuan Cheng, Ting Chen, Xinyang Yi, Dong Lin, Lichan Hong, and Ed H Chi. 2020. Learning Multi-granular Quantized Embeddings for Large-Vocab Categorical Features in Recommender Systems. In *Companion Proceedings of the Web Conference 2020*.

[26] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[27] Tingting Liang, Congying Xia, Yuyu Yin, and Philip S Yu. 2020. Joint Training Capsule Network for Cold Start Recommendation. In *SIGIR*.

[28] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003).

[29] Siyi Liu and Yujia Zheng. 2020. Long-tail Session-based Recommendation. In *Fourteenth ACM Conference on Recommender Systems*.

[30] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[31] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2020. Off-policy Learning in Two-stage Recommender Systems. In *Proceedings of The Web Conference 2020*.

[32] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems* (2019).

[33] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2020. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314* (2020).

[34] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* (2020).

[35] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[36] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*.

[37] Zhimao Peng, Zechao Li, Junge Zhang, Yan Li, Guo-Jun Qi, and Jinhui Tang. 2019. Few-shot image recognition with knowledge transfer. In *Proceedings of the IEEE International Conference on Computer Vision*.

[38] Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. (2016).

[39] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*.

[40] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*.

[41] K Soumya George and Shibily Joseph. 2014. Text classification by augmenting bag of words (BOW) representation with co-occurrence feature. *IOSR J. Comput. Eng* 16, 1 (2014).

[42] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[43] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

[44] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research* 15, 1 (2014).

[45] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in neural information processing systems*.

[46] Ruoxi Wang, Zhe Zhao, Xinyang Yi, Ji Yang, Derek Zhiyuan Cheng, Lichan Hong, Steve Tjoa, Jieqi Kang, Evan Ettinger, and H Chi. 2019. Improving Relevance Prediction with Transfer Learning in Large-scale Retrieval Systems. (2019).

[47] Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. 2019. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE international conference on computer vision*.

[48] Yu-Xiong Wang and Martial Hebert. 2016. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*. Springer.

[49] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Advances in Neural Information Processing Systems*.

[50] Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. *arXiv preprint arXiv:1802.03796* (2018).

[51] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *Companion Proceedings of the Web Conference 2020*.

[52] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*.

[53] Fengjing Yin, Zhenwen Wang, Wentang Tan, and Weidong Xiao. 2012. Sparsity-tolerated algorithm with missing value recovering in user-based collaborative filtering recommendation. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE* 10, 15 (2012).

[54] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *arXiv preprint arXiv:1205.6700* (2012).

[55] Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2019. Addressing the item cold-start problem by attribute-driven active learning. *IEEE Transactions on Knowledge and Data Engineering* 32, 4 (2019).