Shared Neural Item Representations for Completely Cold Start Problem

Ramin Raziperchikolaei
Rakuten, Inc.
San Mateo, CA, USA
ramin.raziperchikola@rakuten.com

Guannan Liang
Rakuten, Inc.
San Mateo, CA, USA
guannan.liang@rakuten.com

Young-joo Chung
Rakuten, Inc.
San Mateo, CA, USA
youngjoo.chung@rakuten.com

ABSTRACT

Neural networks have become popular recently in recommendation systems to extract user and item representations. Most previous works follow a two-branch setting, where user and item networks learn user and item representations in the first and second branches, respectively. In the item cold-start problem, where the usage patterns of the items do not exist, the user network uses ID/interaction vector as the input and the item network uses the item side information (content) as the input. In this paper, we will show that by using this structure, two representations are learned for each item in the training set; one is the output of the item network and the other one is hidden inside the user network and is used for learning user representations. Learning two representations makes training slower and optimization more difficult. We propose to unify the two representations and only use the one generated by the item network. Also, we will show how attention mechanisms fit in our setting and how they can improve the quality of the representations. Our results on public and real-world datasets show that our approach converges faster, achieves higher recall in fewer iterations, and is more robust to the changes in the number of training samples compared to the previous works.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

item cold-start, hybrid recommendation systems, neural networks

ACM Reference Format:
Ramin Raziperchikolaei, Guannan Liang, and Young-joo Chung. 2021. Shared Neural Item Representations for Completely Cold Start Problem. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3460231.3474228

1 INTRODUCTION

Recommendation systems (RSs) help users find what they are interested in from a large set of items and products. This is usually done by designing an algorithm that takes a user and an item and predicts their similarity or matching score.

RSs utilize various input sources to fulfill their task. One important source of information is the previous feedback (interaction) of the users on items, which could be explicit (such as rating) or implicit (such as clicked or purchased). Another source of information is the user and item side information; the user side information includes age, gender, occupation, etc., and the item side information includes title, price, genre, etc [20].

The two main approaches in RSs are collaborative filtering (CF) [15, 26] and content-based filtering (CBF). CF only uses the feedback history to predict the unseen interactions between the users and items. Matrix factorization (MF) [15] is the most popular method in the CF approaches, which maps users and items into a latent space and estimates their similarity by the dot product. The performance of the CF methods drops when the feedback is sparse. CF becomes inapplicable to the cold-start problems, where we need to predict scores for a new user or item without any previous feedback. CBF uses the user and item's side information to make a prediction. The advantage of CBF is that it is applicable as long as the side information is available, and thus can be used for the cold-start problems. When the feedback is available, however, it is known that CF methods usually perform better than CBF methods [24].

Hybrid methods have been proposed to overcome the limitations of both CF and CBF. They use all the available sources of information by mapping the side information and the feedback to separate low-dimensional representations and combine them to predict the final interaction [6, 19, 25, 33]. These methods have shown their effectiveness for cold-start recommendations. *In this paper, we propose a new hybrid method to solve complete item-cold start problems.* In this setting, the test items are completely cold-start, which means that there is no feedback history for them and we only have access to their side information.In this paper, we focus on predicting the implicit feedback of the users on the cold-start items. Note that our method can easily be modified to predict the explicit feedback or to solve the user cold-start problem.

Hybrid cold-start models have at least two separate branches [6, 17, 23, 29]. In the first branch, the item side information is used as the input and mapped to a low-dimensional representation. In the second branch, either user ID [23] or user interaction vector [6, 25, 34] is used as the input to a model which generates user representations. After neural networks became widely used for representation learning, hybrid RSs also adopted neural networks to generate user/item representations [6, 17, 23, 25, 29, 33]. Fig. 1 shows the general structure for hybrid cold-start models.

In this paper, we empirically show that the model with the user interaction vector as the input achieves better recall much faster

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09…$15.00
https://doi.org/10.1145/3460231.3474228

(in fewer training iterations) than the model with the user ID as the input. We investigate this and explain the reason, which comes from the hidden item embedding matrix inside the model with the interaction vector as the input. The hidden embeddings improve the quality of the user representation at initialization.

We propose a method that can achieve even better recall than the model with the interaction vector as the input using fewer training iterations. Our method shares the neural item representations with the hidden item embeddings. By doing so, the item embeddings become a function of their side information and their similarity will be preserved (to some extent) even at the initialization. This will also improve the quality of the user representations and will lead to better results in fewer iterations.

Furthermore, by reformulating our main objective function, we show how the initial user representation is a summation of the user's neighbor representations. We extend our idea by using the attention mechanism to weigh the neighbor representations differently based on their similarity to the test items.

We have conducted extensive experiments on the public and real-world datasets. These experiments confirm the advantages of our approach over the previous works, which include: 1) achieving faster convergence and better recall in a fewer number of training iterations, 2) performing well even when using a smaller number of training data, and 3) improving the performance with the attention mechanism.

2 PREVIOUS WORKS

2.1 Matrix factorization and neighborhood-based methods

Matrix factorization and neighbor-based methods are the two major collaborative filtering approaches. Several matrix factorization techniques have been proposed, including singular value decomposition, probabilistic latent semantic analysis [13] and probabilistic matrix factorization [22]. Neighbor-based methods [2] predict users' unseen preference on items based on the known preferences on the similar items. Takács et al. [26] investigated various regularization scenarios for the matrix factorization approach and then introduced two neighbor-based approaches. Koren [14] built a combined model, which smoothly merges matrix factorization and neighborhood models.

Due to the immense success of deep neural networks, many neural network-based recommender systems have been proposed. He et al. [12] proposed neural collaborative filtering (NCF) that replaces the dot product in collaborative filtering with the deep neural networks. Later, its variants were studied, which take either user/item interaction vectors [7, 32] or user/item ID [10, 12] as inputs. However, CF and NCF require interaction matrices for training and only work in the warm-start setting.

2.2 Attentions in recommendation systems

Attention mechanisms, which are shown to improve the performance of natural language processing and computer vision tasks, have been studied in recommendation tasks as well. The goal of attention mechanisms is to focus on the specific input source which contributes most to the recommendations. Attentive collaborative filtering (ACF) [4] was proposed for multimedia recommendations. ACF uses component-level attention to give more weight to the informative regions in an image. He et al. [11] proposed neural attentive item similarity (NAIS), which uses the attention mechanism to distinguish the important items in the user's past interactions. However, NAIS uses item IDs and item embeddings, which makes its approach inapplicable to the item cold-start problems. Shi et al. [23] used the attention mechanism to determine the importance of different information sources in making prediction. This approach is inapplicable in our setting since there is only one source of information per user and item at test time.

2.3 Recommendation systems for cold-start problems

Various hybrid models have been proposed to tackle the cold-start problem. Collaborative topic regression (CTR) [30] and its variants [9, 31] have been one of the popular structures. Based on the availability of feedback information, CTR interpolates between content-based representations generated from side information by latent Dirichlet allocation (LDA) [3] and feedback-based representations generated from interaction matrices by weighted matrix factorization (WMF).

Recent hybrid methods utilize neural networks to learn representations from side information. To solve the cold-start problem in the music recommendation system, DeepMusic [27] creates multiple embeddings based on the contents (e.g., artist biographies and audio spectrogram) using neural networks. Li et al. [17] combined probabilistic matrix factorization (PMF) with the marginalized denoising autoencoders (mDA) [5]. Dong et al. [6] combined stacked denoising autoencoder (aSDAE) [28] and matrix factorization to integrate side information. Zhang et al. [33] merged contractive autoencoders [21] and the latent factor models like SVD. Li et al. [18] adopted independent encoder-decoder architectures for different data sources, such as numerical, categorical, and sequential data.

Attentional collaborate and content models (ACCM) [23] applies attention mechanisms to hybrid recommender systems. The attention mechanism is used to adjust the importance of the input sources. For cold-start items whose feedback information is missing, ACCM pays more attention to the item's side information to make predictions.

Some works focus on integrating side information and the preference representations generated by the feedback information. Dropoutnet [29] takes preference representation and side information as the inputs for items. For a cold-start item, the values of preference representations are zero since past interactions are missing. CB2CF [1] learns a mapping function from side information to the preference representation. Both Dropoutnet and CB2CF need to first learn preference representations by applying a matrix factorization algorithm to the interaction matrix. As a result, the recommendation performances of Dropoutnet and CB2CF are upper-bounded by the performance of matrix factorization, while the performance of a matrix factorization algorithm heavily depends on the number of non-zero elements in the matrix. It means that, with an extremely sparse interaction matrix, Dropoutnet and CB2CF both have poor recommendation performance.

Recently, researchers have explores meta-learning for item cold-start problems. Based on the MAML framework [8], Meta-Learned

User Preference Estimator for Cold-Start Recommendation (MELU) has been proposed in [16]. Treating each item as a task of meta-learning, MELU can rapidly adapt the learned recommender system to new items with a small number of new interactions.

3 NEURAL NETWORK-BASED METHODS IN THE COLD-START PROBLEM

In this section, we review how the previous neural network-based methods work in the cold-start setting. Let us introduce the notations first. We denote the sparse interaction matrix by $\mathbf{R} \in \mathbb{R}^{m \times n}$, where m and n are the number of users and items, respectively. In the interaction matrix, $R_{jk} = 1$ means user j interacted with (purchased) item k, and $R_{jk} = 0$ means the interaction is unknown. The s-dimensional side information of all the n items are denoted by $\mathbf{X} \in \mathbb{R}^{n \times s}$. The ith row of a matrix $\mathbf{H}$ is shown by $\mathbf{H}_{i,:}$ and the jth column is shown by $\mathbf{H}_{:,j}$.

In our setting, there is no user side information and there is no item feedback at test time. In this setting, the previous cold-start or hybrid methods will have a two-branches structure as shown in Fig. 1 [6, 17, 23, 29, 33]. The user representation is created in the user branch and the item representation is created in the item branch. A prediction function takes the two representations and maps them to a number which shows the interest of the user in that item. In the following, we explain each part of this structure in more detail.

For the items, a multi-layer perceptron (MLP), denoted by $\mathbf{g}^i()$, takes the item side information of the kth item $\mathbf{X}_{k,:}$ and maps it to a d^i-dimensional representation $\mathbf{z}^i_k$, i.e., $\mathbf{z}^i_k = \mathbf{g}^i(\mathbf{X}_{k,:}) \in \mathbb{R}^{d^i}$.

For the users, the input source is either their IDs or interaction vectors. In the case of ID as the input, an embedding layer is used to extract the jth user representation from the jth row of an embedding matrix. First, the user ID is converted to a one-hot-encoding vector of size m (number of users). For user j, this vector is denoted by $\mathbf{I}^u_j \in \mathbb{R}^m$, where the jth entry of $\mathbf{I}^u_j$ is 1 and the rest are 0. Then, this vector is multiplied by a user embedding matrix $\mathbf{E}^u \in \mathbb{R}^{m \times d^u}$ to give us the d^u-dimensional representation vector $\mathbf{z}^u_j$, i.e., $\mathbf{z}^u_j = \mathbf{I}^u_j \mathbf{E}^u \in \mathbb{R}^{d^u}$.

In the case of the interaction vector as the input, for the jth user, a multi-layer perceptron takes the jth row of the interaction matrix and maps it to a d^u-dimensional representation:

$$\mathbf{z}^u_j = \mathbf{g}^u(\mathbf{R}_{j,:}) = \sigma(\dots \sigma(\sigma(\mathbf{R}_{j,:}\mathbf{W}^u_1)\mathbf{W}^u_2)\dots \mathbf{W}^u_L), \tag{1}$$

where $\sigma()$ is an activation function and $\mathbf{W}^u_l$ is the weight matrix of the lth layer.

Different kinds of loss functions have been used in the literature to train the model and learn the representations. Here are two examples:

$$l_{\text{contrastive}} = \sum_{j,k \in S^+} ||\mathbf{z}^u_j - \mathbf{z}^i_k||^2 + \lambda \sum_{j,k \in S^-} \max(0, m_d - ||\mathbf{z}^u_j - \mathbf{z}^i_k||)^2$$

$$l_{\text{dot-mse}} = \sum_{j,k \in S^+ \cup S^-} ||(\mathbf{z}^u_j)^T \mathbf{z}^i_k - R_{ij}||^2 \tag{2}$$

where m_d is a margin. S^+ is a set of positive pairs of users and item, where $(j,k) \in S^+$ if user j interacted with the item k (with $R_{jk} = 1$). S^- is a set of negative pairs of users and items, which are randomly selected from the non-positive pairs (with $R_{jk} = 0$). Contrastive loss tries to make the distance between the representation of the similar/dissimilar users and items small/big. In the dot-mse loss function, the interaction value (0 or 1) is estimated by the dot product of the representations.

In the above two loss functions, the assumption is that the user and item representations have the same dimension, i.e., $d^u = d^i$. To avoid such an assumption, one can concatenate the two representations and use an MLP to predict the interaction score. The focus of this paper is not on the exact form of the loss function and we use the ones in Eq. (2)

4 OUR PROPOSED METHOD: SHARED NEURAL ITEM REPRESENTATION

Our approach is motivated by the fact that the model with the interaction vector as the input achieves better results in a fewer number of iterations (mini-batches in training) than the model with the ID as the input. In Fig. 2, "nonshared_ID" refers to the model with the ID as the user input, and "nonshared_interaction" refers to the model with the interaction vector as the user input. Both models use the side information as the item input. We show the recall and training error of the methods on Ichiba1M. The details of this experiment and the dataset can be found in our experimental results section 5.2. Note that the item MLP and the loss function are the same in the two experiments and the only difference is in their user representations.

We can see in Fig. 2 that by using the interaction vector as the input the loss decreases and recall increases much faster than using ID as the input. This happens because, even at the initialization and before training the networks, "nonshared_interaction" generates better user representations than "nonshared_ID".

To better understand the difference between the two inputs, we further investigated the quality of the representations. We selected a set of 20,000 users randomly and computed their representations. Then, we computed the pairwise distances between the representations and computed their probability density functions. Fig. 3 (a) shows the probability density function at initialization, where we compute the user representations and distances without training the networks. Fig. 3 (b) shows the probability densities at convergence, where the representations are obtained from the trained models with the maximum recall. Fig. 3 (c) is a magnified version of the the Fig. 3 (b).

As we can see in Fig. 3 (a), the probability density function of the "nonshared_ID" as the input (red curve) is similar to the density function of a normal distribution with a very small variance. In other words, all the user representations are almost the same. We can also see that "nonshared_interaction" (blue curve) has a larger variance, which means that some users are more similar to each other than other users. In Fig. 3 (b) and (c), we can see that the density functions become very similar at the convergence.

To understand why using interaction vector as the input leads to a better user representation at initialization, let us take a deeper look at the user MLP in Eq. (2). First, note that the input vector is n dimensional and the weight matrix of the first layer is $\mathbf{W}^u_1 \in \mathbb{R}^{n \times p}$. In other words, $\mathbf{W}^u_1$ contains a p-dimensional vector for each of the n training items. We call this weight matrix a *hidden item embedding matrix*. Second, note that most entries of the jth user interaction vector $\mathbf{R}_{j,:}$ is 0. The non-zero elements correspond to the items

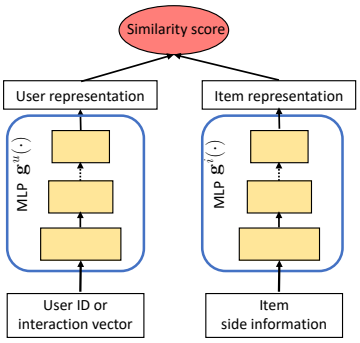

Figure 1: Two-branch structure of the neural network-based methods in the cold-start problem.

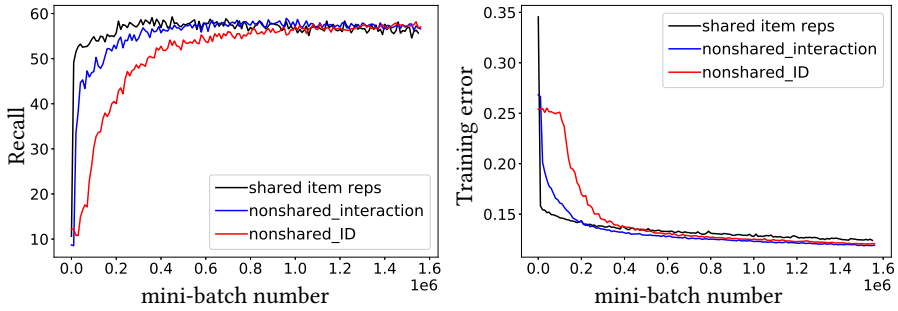

Figure 2: Recall and training error of the methods at different mini-batches in training. Recall is computed on the validation set.

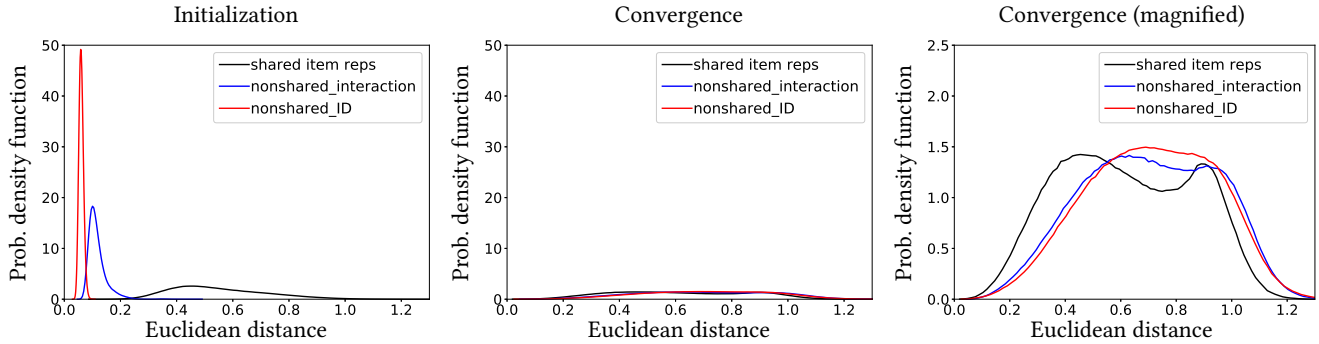

Figure 3: We randomly selected a set of 20,000 users, computed their representations at initialization and convergence, and then computed the pairwise distance between the representations. We show the probability density function of the pairwise distances.

that user j interacted with. Therefore, the output of the first layer (before applying the activation function) is a summation of the hidden embeddings of the items that the user has interacted with.

4.1 Sharing neural item representations with the hidden item embeddings

We aim to to achieve better recall than the model with the interaction vector as the input in fewer iterations. Recall that the main advantage of the "nonshared_interaction" is having better user representations even at the initialization. This is achieved with the help of the hidden item embedding matrix in the user MLP, which puts the users with the same purchase history closer to each other even before training the networks. However, the hidden item embedding layer is initialized randomly and does not preserve the similarity between the items, which significantly affects the similarity between the users' representations.

Our main idea is to make the hidden item embedding of the user MLP be a function of the items' side information. This way, the similarity between the user representations will be preserved better from the initialization to the convergence.

Our proposed model uses the same model and objective function as the previous works with the interaction vector as the inputs. The only difference is that *we share the item representations from the item model with the hidden item embedding of the user model.* Here is our objective function:

$$\sum_{j,k \in S^+ \cup S^-} ||(\mathbf{z}_j^u)^T \mathbf{z}_k^i - R_{ij}||^2 \quad \text{s.t.}$$
$$\mathbf{z}_j^u = \mathbf{g}^u(\mathbf{R}_{j,:}) = \sigma(\dots\sigma(\sigma(\mathbf{R}_{j,:}\mathbf{W}_1^u)\mathbf{W}_2^u)\dots\mathbf{W}_L^u),$$
$$\mathbf{W}_1^u = \mathbf{g}^i(\mathbf{X}), \quad \mathbf{z}_k^i = \mathbf{g}^i(\mathbf{X}_{k,:}). \tag{3}$$

The above loss function is minimized over the parameters of the $\mathbf{g}^i()$ and $\mathbf{g}^u()$, where $\mathbf{g}^u()$ is the user MLP with the parameters $[\mathbf{W}_2^u, \dots, \mathbf{W}_L^u]$. Note that $\mathbf{W}_1^u$ is the hidden item embedding matrix, which is not a parameter in our formulation. This matrix is replaced by the output of the item MLP.

In Fig. 2, we can see that our "shared item reps" model (black curve) achieves a higher recall using a smaller number of iterations. Let us take a look at the density function of the users at initialization in Fig. 3 (a). As we can see, the density function of our "shared item reps" model is completely different from the other methods and it is very close to the density at the convergence.

Note that when we share the item representation with the hidden item embeddings, we must ensure that both embeddings and

representations have the same dimension. To avoid this, and let each of them have its own dimension, we can define a new MLP $\mathbf{h}()$ and use its output as the hidden item embeddings instead of the output of $\mathbf{g}^i()$, i.e., $\mathbf{W}_1^u = \mathbf{h}(\mathbf{X})$. In this case, we have two separate MLPs to map side information to the low-dimensional space, which makes the model more flexible and powerful. However, the extra MLP $\mathbf{h}()$ adds a lot of new parameters to the loss function and makes optimization difficult. As our main motivation is to achieve better results faster, we focus on using the shared model.

4.2 Faster training with a simpler formulation

We optimize the objective function of Eq. (3) with the mini-batch gradient descent. Each mini-batch contains several pairs of users and items. Given a mini-batch, we first compute the representation of all the items and set $\mathbf{W}_1^u = \mathbf{g}^i(\mathbf{X})$. Then, we compute the user and item representations and the loss value using Eq. (3). Computing the representation of all the items at each mini-batch is time-consuming and unnecessary. In the following, we rewrite the objective function of Eq. (3) and show how we can significantly reduce this computation.

Let us define the neighbor set of user j as the set of items that have been purchased by that user, denoted by $N_j = \{k | R_{jk} \neq 0\}$. For the jth user, we can rewrite the output of the first layer of the user MLP as $\sigma(\mathbf{y}_j)$, where $\mathbf{y}_j$ is defined as follows:

$$\mathbf{y}_j = \mathbf{R}_{j,:}\mathbf{W}_1^u = \mathbf{R}_{j,:}\mathbf{g}^i(\mathbf{X}) = \sum_{p \in N_j} \mathbf{g}^i(\mathbf{X}_{p,:}). \tag{4}$$

The above equation shows that for each user j, its initial representation (i.e., output of the first layer) is computed as the sum of the representations of its neighboring items N_j followed by a nonlinear activation. Therefore, in each mini-batch, we compute the user representations as follows: we get N_j for each user j in that mini-batch, apply $\mathbf{g}^i()$ to each item in N_j, use Eq. (4) to compute its initial representation, and use $\mathbf{W}_2^u$ to $\mathbf{W}_L^u$ in Eq. (3) to compute the representation $\mathbf{z}_j^u$.

In real-world datasets, each user has a small number of neighbors. Since the size of the mini-batches is also small, the above strategy computes representations of a much smaller number of items compared to the total number of items.

4.3 Attention mechanism in learning user representations

As we explained in Eq. (4), for each user j, its initial representation (i.e., output of the first layer of the user MLP before applying the activation function) , is the summation of the representations of the items that the user interacted with. We can see in Eq. (4) that all the items in the neighbor set N_j contribute equally (have the same weight).

Recall that we never compute the user representation in isolation. Our approach always takes a pair of user j and item k, and then maps their representations either close or far away from each other. It is therefore reasonable to consider item k when we compute the jth user representation. Our idea is to make the initial user representation a weighted sum of its neighbors' item representations, where the weights are computed based on the similarity to the kth item. We define our new objective function as follows:

$$\sum_{j,k \in S^+ \cup S^-} ||(\mathbf{z}_j^u)^T \mathbf{z}_k^i - R_{ij}||^2 \quad \text{s.t.}$$

$$\mathbf{z}_j^u = \mathbf{g}^u(\sigma(\mathbf{y}_j)), \quad \mathbf{y}_j = \sum_{p \in N_j} \alpha_{pk} \mathbf{g}^i(\mathbf{X}_{p,:}) \quad \mathbf{z}_k^i = \mathbf{g}^i(\mathbf{X}_{k,:}) \tag{5}$$

where $\mathbf{y}_j$ is the initial representation of the jth user. The key in the above equation is α_{pk}, which gives weight to the representation of the pth item based on its similarity to the kth item. In this paper, we apply cosine, dot product, and general attention mechanisms to learn the weights. We first compute $\hat{\alpha}_{pk}$ as follows:

$$\text{dot: } \hat{\alpha}_{pk} = (\mathbf{z}_p^i)^T \mathbf{z}_k^i \quad \text{general: } \hat{\alpha}_{pk} = (\mathbf{z}_p^i)^T \mathbf{W}_\alpha \mathbf{z}_k^i$$

$$\text{cosine: } \hat{\alpha}_{pk} = \frac{(\mathbf{z}_p^i)^T \mathbf{z}_k^i}{||\mathbf{z}_p^i|| ||\mathbf{z}_k^i||}, \tag{6}$$

where $\mathbf{W}_\alpha$ is a learnable matrix in the general attention. The final weights are achieved by applying a softmax function:

$$\alpha_{pk} = \frac{\exp(\hat{\alpha}_{pk})}{\sum_{p' \in N_j} \hat{\alpha}_{p'k}}. \tag{7}$$

5 EXPERIMENTS

5.1 Experimental setup

5.1.1 Datasets. We have conducted experiments on three datasets:

(1) **CiteULike**. In this dataset, each user has a set of saved articles and the goal is to recommend new (cold-start) articles to the users. We use the subset provided by [29], which contains 5,551 users, 16,980 articles, and 204,986 user-article pairs. The interaction matrix $\mathbf{R}$ is 99.8% sparse and $R_{jk} = 1$ means user j saved article k. A set of 3,396 articles are removed from the training data and used as the test cold-start articles.

(2) **Ichiba1m**. This dataset contains around 1 million purchases from a specific category of the Rakuten Ichiba [1]. There are around 265,000 items, 100,000 users, and 200 cold-start items. The dataset is 99.996% sparse.

(3) **Ichiba20m**. This dataset contains 20 million purchases from 40 different categories of the Rakuten Ichiba. There are around 5 million items, 7.5 million users, and 24,000 cold-start items. The dataset is 99.99994% sparse.

5.1.2 Item side information. The side information is the input to the item MLP. In the CiteULike dataset, we follow the approach of [29]. The side information of the articles includes their abstracts and titles. A vocabulary of the top 8,000 words is selected by tf-idf and used as the articles' features.

Rakuten Ichiba's category taxonomy has five levels, from broad genres (the first level, e.g., shoes) to specific genres (the fifth level, e.g. running shoes).

In Ichiba1m and Ichiba20m, we use second-level categories, third-level categories, and descriptions as the item side information. The categories are converted into one-hot-encoding. For the items' descriptions, we tokenize them with MeCab[2], then convert the first 50 tokens into a feature vectors using bag-of-words.

[1] https://rit.rakuten.co.jp/data_release/
[2] https://pypi.org/project/mecab-python3/

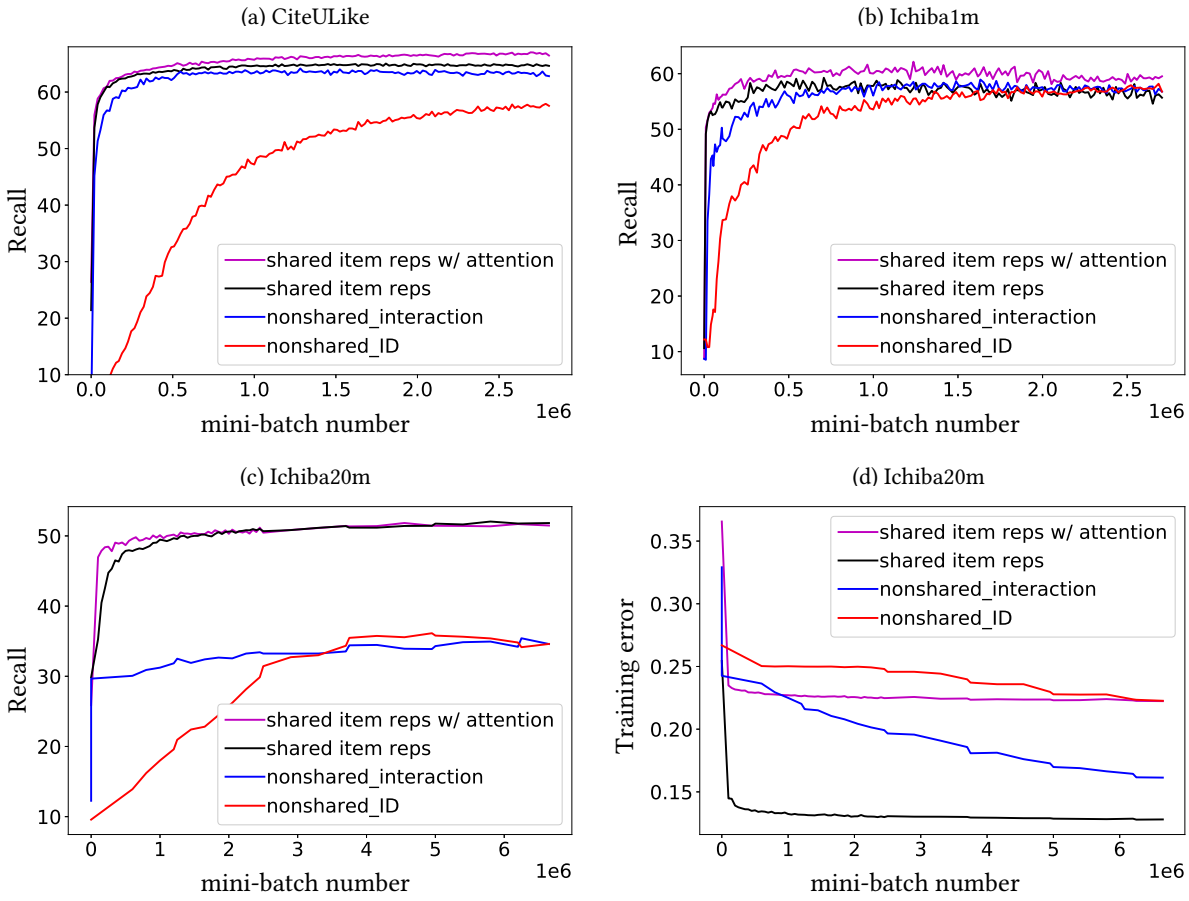

Figure 4: In (a), (b), and (c), we report recall on the validation set at different mini-batch numbers as as we train the models in CiteULike, Ichiba1m, and Ichiba20m datasets. In (d), we report the training error in the Ichiba20m dataset.

In Ichiba1m, there are 14 second-level categories, 150 third-level categories, and 68,000 tokens in the vocabulary. In Ichiba20m, there are 3,800 second-level categories, 540 third-level categories, and 719,000 tokens in the vocabulary.

5.1.3 Evaluation metrics. We report average recall in all three datasets to evaluate the methods. The recall for a single user or item is defined as:

$$\text{Recall} = \frac{|\text{relevant set}| \cap |\text{retrieved set}|}{|\text{relevant set}|}. \tag{8}$$

The relevant and retrieved sets are defined differently in Ichiba and CiteULike datasets. Note that in all RSs algorithms, we can find similarity scores between any pair of users and items and find top similar ones based on the scores.

In Ichiba1m and Ichiba20m datasets, the goal is to find interested users for a newly released (cold-start) item. For each cold-start item (not appeared in our training set), the relevant set contains users who purchased that item. The retrieved set contains the top K users with the maximum similarity to the cold-start item.

In CiteULike dataset, we follow [29] in computing the recall. The goal in this dataset is to assign a set of cold-start articles to each user. Therefore, for each user, the relevant set is defined as the set of cold-start articles that have been saved by that user. The retrieved set contains the top K cold-start articles (with the maximum similarity) for each user.

5.1.4 Implementation details. We implemented the methods using Keras with TensorFlow 2 backend. We ran all the experiments on a 12 GB GPU. We set the mini-batch size to 32 and the maximum number of epochs to 50. We use SGD in all experiments and pick the best learning rate from 10^{-1} to 10^{-4}. The learning rate of our shared model and its variants is 0.01 in the Ichiba1m and Ichiba20m datasets and 0.001 in the CiteULike dataset.

The user and item embedding sizes are fixed to 100 in all experiments. Unless otherwise stated, the user and item MLPs have 3 fully connected layers. In Ichiba20m and Ichiba1m, we use the contrastive loss and in the CiteULike we use the dot-mse of Eq. (2) as the loss function.

We set $K = 100$ in computing the recall in all experiments. We report the recall several times in each epoch during the training process to show which method achieves the highest recall faster. Computing the retrieved set over all the users this way is very time-consuming in Ichiba20m and Ichiba1m. Therefore, in these

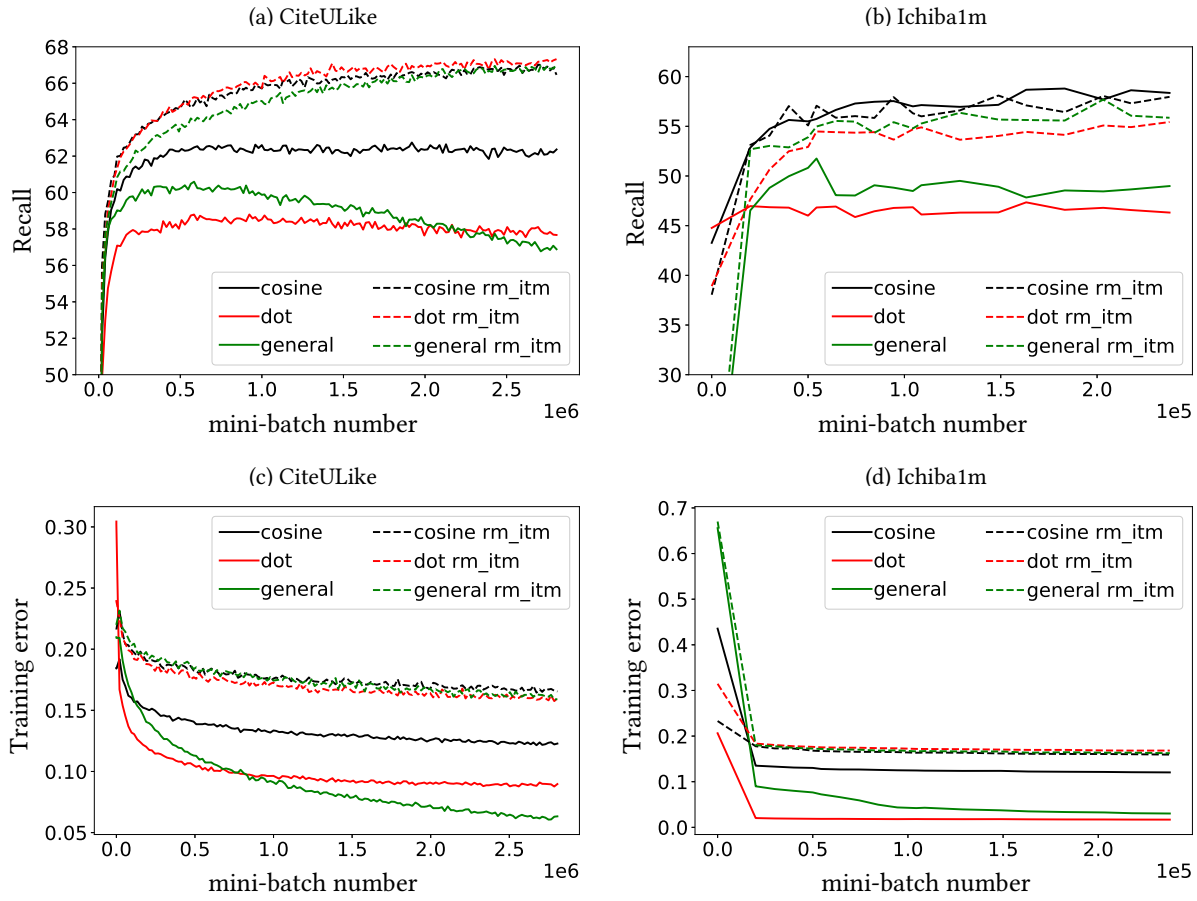

Figure 5: We report recall (top row) and training error (bottom row) at different mini-batch numbers during the training. The recall and error are reported on the CiteULike (left panel) and Ichiba1m (right panel).

two datasets, the retrieved set for each cold-start item is computed over a subset of the users, which contains: 1) all the buyers and 2) 1,000 randomly selected users from the non-buyers. We verified that by using all the users or a subset of them in computing the recall, the order of the methods remains the same.

5.2 Experimental results

5.2.1 Sharing item representations leads to better recall in fewer training iterations. In Fig. 4 (a), (b), and (c), we report recall on the validation set at different batch numbers as we train the models. We can see the same pattern in all three datasets. The "non-shared_interaction" always achieves better recall faster than "non-shared_ID" because of having better user representations at the initialization. Our "shared item reps" model, which shares the item representations with the hidden item embeddings, achieves a higher recall even faster than "nonshared_interaction".

We can see that the advantage of the "shared item reps" is more significant in the Ichiba20m dataset. This is because Ichiba20m has a lot more items and users than the rest of the datasets. Note that the number of parameters of the non-shared models almost linearly increases with the number of users and items. In our case, since the representations are shared, the number of parameters is significantly smaller. This helps the shared model to have easier optimization and better generalization on bigger datasets.

5.2.2 Attention mechanism improves the performance. In Fig. 4, we have added the "shared item reps" model with the (cosine) attention mechanism to the plots. We can see that its recall is significantly better than non-shared models and slightly better than the "shared item reps" model without the attention.

To understand how the attention mechanism achieves such a good result, we have plotted the training error of the methods in Fig. 4 (d). We can see that the training error of the attention model is not better than the rest of the methods. This means that the success of the attention mechanism is because of its complex process at test time. Recall that the non-attention models can compute and store the user representations when the training is done. In other words, the user representations are fixed after the training, no matter what the test item is. When we use the attention mechanism, the user representations are not fixed and updated based on the test item. This is because the test items define the weights in combining the neighbor item representations in Eq. (5). This more complex and

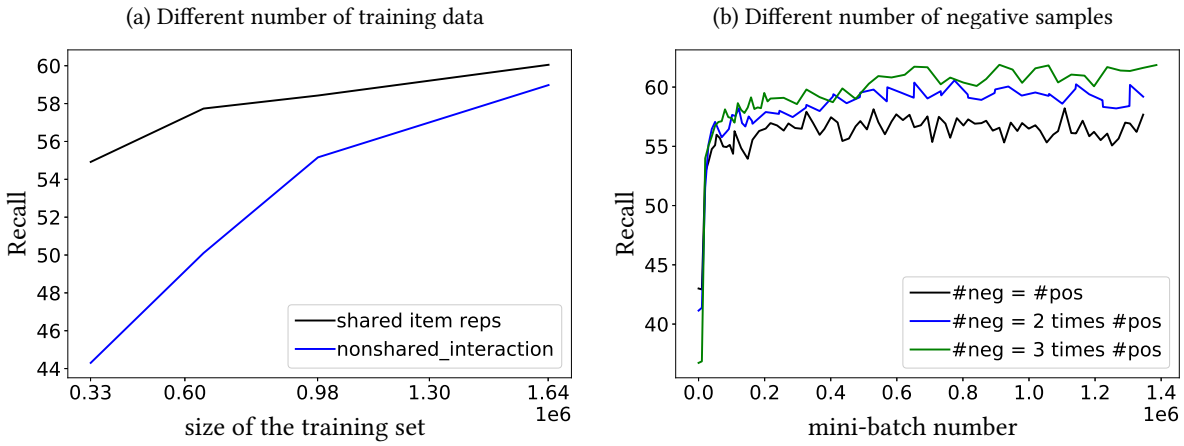

Figure 6: Both plots belong to the Ichiba1m dataset. *Left:* We change the size of the training set from 330K to 1.6M and train the models. The "shared item reps" model needs less data than the "nonshared_interatcion" model to achieve reasonable results. *Right:* We train the "shared item reps" model with a different number of negative samples and report the recall. The performance of the model improves as we increase the number of negative samples.

time-consuming process at the test time improves the recall, even without decreasing the training error significantly.

5.2.3 Different attention mechanisms. In Fig. 5, we compare the attention mechanisms in CiteULike (left) and Ichiba1m (right). There are two versions for each attention mechanism: with and without removing the item. Let us explain what they mean.

Consider a positive pair in the training set, e.g., user j and item k with $R_{jk} = 1$. Since $R_{jk} = 1$, the neighbor set of the user j contains item k, i.e., $k \in N_j$. As we can see in Eq. (5), we have to compute the similarity between item k and all the items in N_j. This might lead to overfitting since N_j includes k and the similarity between item k and itself could become very large and dominate other similarity values. For this reason, we have implemented two variants of each attention method, one with all the items in the neighbor set and one with removing the item from the neighbor set. In Fig. 5 , when we remove the item from the neighbor set of the user, we put "rm_itm" besides the type of the attention mechanism. Note that removing the item only happens at training. During the test, we use all the items in the neighbor set.

Let us take a look at the bottom row of Fig. 5, where we report the training errors. In both datasets, the models achieve better training errors without removing the items, as expected. It is remarkable that the cosine attention without removing the item stays in between the other models in both datasets. This is mainly because of the normalization (dividing by the norm) that happens in computing the cosine similarity. This normalization does not let the item representations become too large, acts as a regularizer, and does not let the training error become as low as the dot and general mechanisms.

If we look at the recall in the top row, we can see that the dot and general mechanisms (with all the items) are clearly overfitting. All the attention mechanisms with the "rm_itm" strategy achieve higher recall. The only exception is the cosine in the Ichiba1m, which performs almost as well as other methods with the "rm_itm". This means that *at test time*, the cold-start items have high similarity to at least one item in the neighbor set of interested users. Cosine attention takes advantage of this property of the Ichiba1m, avoids overfitting using the normalization factor, and achieves the maximum recall.

5.2.4 Sharing item representations is more crucial when models are trained on smaller subsets of training data. In the real-world datasets, we have access to hundreds of millions of purchases (user-item pairs). Training the model on all the data can take a very long time. Therefore, the model that is more robust to the change in number of training data is more applicable in real-world scenarios.

In Fig. 6 (a), we designed an experiment on the Ichiba1m to compare the sensitivity of the models to the number of training data. We trained the models on a subset of the Ichiba1m dataset, where the size of the subsets is between 330K to 1.6M. The test cold-start items were the same in all experiments.

The recall of both methods in Fig. 6 (a) improves as we increase the number of training data. The "shared item reps" model performs significantly better on the smaller subsets. This has two reasons. First, the model with the shared item representations has a smaller number of parameters so it achieves a better generalization using a smaller number of training data. Second, when we decrease the number of training data, the number of "seen" users decreases. The "nonshared_interaction" model has difficulty handling this situation because the representation of the users does not get better if it does not see enough users. In the "shared item reps" model, the representation of the users is a function of item representations. So, as long as it learns meaningful item representations, it can generalize to the unseen users.

5.2.5 Ablation study: number of negative samples and number of layers in the user MLP. In the implicit feedback prediction, we only have access to positive interactions. The negative pairs are sampled randomly from unknown interactions. In Fig. 6 (b), we investigate the importance of increasing the number of negative samples. We

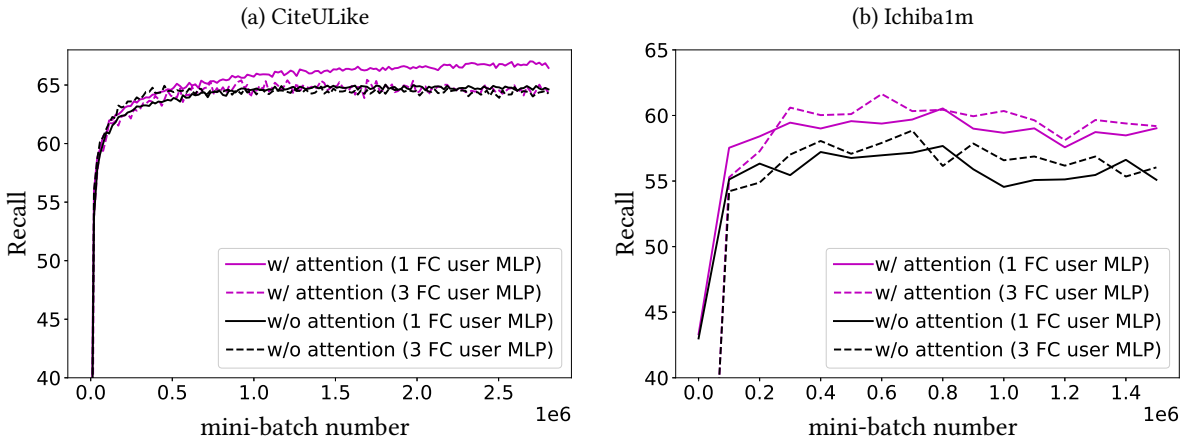

Figure 7: We train our "shared item reps" model with and without the attention mechanism, where the user MLP contains 1 or 3 fully connected layer. We report recall at different mini-batch numbers in training. The deeper user MLPs are helpful in the Ichiba1m dataset.

Table 1: Comparison with the state-of-the-art cold-start methods on CiteULike dataset. The * after the methods' name means we report the results from the DropoutNet paper's table [29]. Our methods outperforms the competitors.

method	Test recall %
shared item reps w/ attention	**66.4**
shared item reps	**65.7**
DN-WMF (DropoutNet, retrained) [29]	65.2
DN-WMF* (DropoutNet) [29]	63.6
ACCM [23]	63.1
DN-CDL* (DropoutNet) [29]	62.9
DeepMusic* [27]	60.1
CTR* [30]	58.9
CDL* [31]	57.3

set the number of negative samples to 1) equal, 2) twice, and 3) three times the number of positive samples. The results in Fig. 6 (b) show that increasing the number of negative samples improves the performance.

In Fig. 7, we investigate how increasing the number of layers of the user MLP affects the performance. In CiteULike dataset, which is smaller and simpler than Ichiba1m, adding layers to the user MLP does not help. But, in Ichiba1m, we can improve the recall by increasing the number of layers.

5.2.6 Comparison with the methods in the literature . In Table 1, we compare our model (with and without the attention mechanism) with several recent cold-start works using CiteULike dataset. As we can see from the table, our methods outperforms the competitors.

The code and the training/test sets of the CiteULike experiment in the DorpoutNet [29] is available online [3]. We use the code and processed data provided in this repository to train our models.

In Table 1, a * after the methods' names means that we use the reported results in the DorpoutNet paper [29]. In DN-WMF (retrained), we trained the DropoutNet model by maximizing the droupout ratio of the item preference vector. We did that to force the model to ignore the preference vector and perform better in the cold-star situation. In ACCM [23], we removed the user side information and the item ID from the model structure to make it work in the complete cold-start situation.

6 CONCLUSION

In this paper, we presented a hybrid method to address the completely item cold-start problem. The two common inputs to learn user representations are the user ID and interaction vectors. We empirically compared these two inputs and showed that using the interaction vector can lead to better results faster. To achieve better user representation and better recall in fewer iterations, we proposed to share the item representations with the hidden item embedding matrix of the user MLP. We also showed how we can use the attention mechanism to further improve the prediction performance. Our experimental results confirmed that our approach outperforms the previous works.

REFERENCES

[1] Oren Barkan, Noam Koenigstein, Eylon Yogev, and Ori Katz. 2019. CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*. https://doi.org/10.1145/3298689.3347038
[2] Robert M. Bell and Yehuda Koren. 2007. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *Proceedings of the 7th IEEE International Conference Data Mining (ICDM)*.
[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research (JMLR)* (2003).
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation

[3] https://github.com/layer6ai-labs/DropoutNet

with Item- and Component-Level Attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. https://doi.org/10.1145/3077136.3080797

[5] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

[6] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. 2017. A hybrid collaborative filtering model with deep structure for recommender systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

[7] Zhi-Hong Dong, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and Philip S Yu. 2019. DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

[9] Prem Gopalan, Jake M. Hofman, and David M. Blei. 2015. Scalable recommendation with hierarchical Poisson factorization. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*.

[10] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-Based Neural Collaborative Filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

[11] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018), 2354–2366.

[12] 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 (WWW)*.

[13] Thomas Hofmann. 2004. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems* (2004), 89–115.

[14] Yehuda Koren. 2008. Factorization meets the neighborhood:a Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. https://doi.org/10.1145/1401890.1401944

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/mc.2009.263

[16] 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 and Data Mining (KDD)*.

[17] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*. https://doi.org/10.1145/2806416.2806527

[18] Tianyu Li, Yukun Ma, Jiu Xu, Bjorn Stenger, Chen Liu, and Yu Hirate. 2018. Deep Heterogeneous Autoencoders for Collaborative Filtering. In *Proceedings of the 18th IEEE International Conference Data Mining (ICDM)*.

[19] Ramin Raziperchikolaei, Tianyu Li, and Young joo Chung. 2021. Neural Representations in Hybrid Recommender Systems: Prediction versus Regularization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[20] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 2011. *Recommender Systems Handbook*. Springer US. https://doi.org/10.1007/978-0-387-85820-3

[21] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

[22] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS)*.

[23] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *Proceedings of the 27th ACM International on Conference on Information and Knowledge Management (CIKM)*. https://doi.org/10.1145/3269206.3271710

[24] Malcolm Slaney. 2011. Web-Scale Multimedia Analysis: Does Content Matter? *IEEE Multimedia* (2011), 12–15.

[25] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLSRS)*. https://doi.org/10.1145/2988450.2988456

[26] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2nd ACM Conference on Recommender Systems (RecSys)*. https://doi.org/10.1145/1454008.1454049

[27] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*.

[28] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre antoine Manzagol. 2010. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)* (2010).

[29] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*.

[30] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[31] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. https://doi.org/10.1145/2783258.2783273

[32] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. https://doi.org/10.24963/ijcai.2017/447

[33] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. https://doi.org/10.1145/3077136.3080689

[34] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources. In *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management (CIKM)*. https://doi.org/10.1145/3132847.3132892