Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation

Ziwei Zhu*
Texas A&M University
zhuziwei@tamu.edu

Shahin Sefati
Comcast Applied AI Research Lab
Shahin_Sefati@comcast.com

Parsa Saadatpanah
Comcast Applied AI Research Lab
parsa@cs.umd.edu

James Caverlee
Texas A&M University
caverlee@tamu.edu

ABSTRACT

The cold start problem is a long-standing challenge in recommender systems. That is, how to recommend for new users and new items without any historical interaction record? Recent ML-based approaches have made promising strides versus traditional methods. These ML approaches typically combine both user-item interaction data of existing warm start users and items (as in CF-based methods) with auxiliary information of users and items such as user profiles and item content information (as in content-based methods). However, such approaches face key drawbacks including the error superimposition issue that the auxiliary-to-CF transformation error increases the final recommendation error; the ineffective learning issue that long distance from transformation functions to model output layer leads to ineffective model learning; and the unified transformation issue that applying the same transformation function for different users and items results in poor transformation.

Hence, this paper proposes a novel model designed to overcome these drawbacks while delivering strong cold start performance. Three unique features are: (i) a combined separate-training and joint-training framework to overcome the error superimposition issue and improve model quality; (ii) a Randomized Training mechanism to promote the effectiveness of model learning; and (iii) a Mixture-of-Experts Transformation mechanism to provide 'personalized' transformation functions. Extensive experiments on three datasets show the effectiveness of the proposed model over state-of-the-art alternatives.

CCS CONCEPTS

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

KEYWORDS

collaborative filtering; cold start recommendation; randomized training; mixture-of-experts

*Part of this work performed while interning at Comcast Applied AI Research Lab.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401178

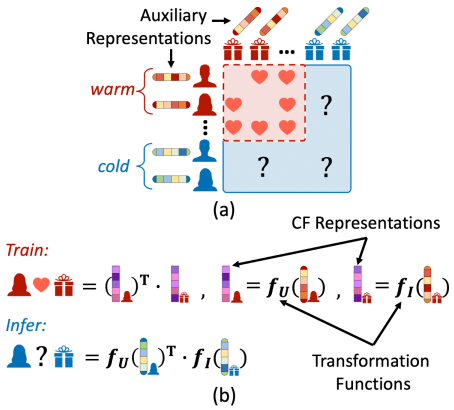


Figure 1: (a) setup of cold start recommendation problem, where both warm and cold users and items have auxiliary representations (such as user profiles and item content); and (b) the main idea of existing cold start recommendation algorithms [7, 17, 28, 35, 36]: learn transformation functions to transform auxiliary representations to CF representations.

ACM Reference Format:
Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401178

1 INTRODUCTION

One longstanding challenge for Collaborative Filtering (CF) based recommendation methods is the *cold start problem*, i.e., to provide recommendations for new users or items who have no historical interaction record. The cold start problem is common in real world applications. For example, 500 hours of new videos are uploaded to YouTube every minute [8], 500,000 new users register in Facebook every day [32], and web/mobile apps face the daily challenge of onboarding new users and subscribers.

To provide recommendations for these new users and items, many content-based methods [13, 21, 34] and heuristic methods have been deployed, e.g., recommending popular items or geographically near items. However, recent research efforts [7, 17, 28, 35, 36] that tackle the cold start problem from the perspective of machine learning have made promising strides. As illustrated in Figure 1a,

these ML-based efforts combine user-item interactions from existing warm start users and items (as in CF-based methods) with auxiliary information from both warm and cold users and items (as in content-based methods). This auxiliary information – be it from user profiles, item descriptions, reviews, and other sources – is often readily available even in the absence of user-item interactions.

Conventional CF models provide recommendations for warm users and items by finding similarities between the CF representations of users and items, which are learned based on existing user-item interactions. In contrast, these ML-based cold start recommendation approaches aim to learn CF representations for cold start users and items lacking historical interactions. The key insight is to learn two transformation functions – one for users and one for items – to transform the auxiliary representations of these new users and items into the CF space. As illustrated in Figure 1b, the two transformation functions (f_U and f_I) are learned from interactions and auxiliary representations of warm users and items; the learned transformation functions are then applied on auxiliary representations of cold start users and items to predict preference scores at inference time. Hence, the fundamental challenge is *how to generate effective transformation functions based on the given auxiliary information and user-item interactions.*

In general, there exist two major categories of algorithms to learn these transformation functions – separate-training methods and joint-training methods. In this paper, we identify three major issues with these existing methods that can impact the quality of the learned transformation functions. These issues motivate our proposed **Heater** framework.

To begin with, separate-training methods [3, 7, 26, 31, 33] separately learn a CF model (by minimizing the Collaborative Filtering error $\mathcal{L}_{CF}$ on the user-item interactions as in conventional CF models for warm start recommendation) and transformation functions to transform auxiliary representations (by minimizing the difference $\mathcal{L}_{trans}$ between transformed auxiliary representations and CF representations from a CF model), either in an end-to-end way or non-end-to-end two-step way [3, 7, 26, 31, 33]. Separate-training methods can fully utilize the user-item interaction data because they apply sophisticated CF models directly on the interaction data. However, they leave one challenge unsolved – the *error superimposition problem*. Due to the separation of learning CF and learning transformation functions in the model objective function, the final cold start recommendation error during inference is the summation of the CF error $\mathcal{L}_{CF}$ and the transformation error $\mathcal{L}_{trans}$. Hence, an increase in either of the two errors will decrease the overall cold start recommendation performance.

On the other hand, joint-training methods [17, 28, 29, 36] fuse CF and transformation functions together (i.e., models input auxiliary representations and output recommendations with CF representations as hidden layers), and train models with the only aim of minimizing the recommendation error $\mathcal{L}_{CF}$ on the warm interaction data. Since joint-training methods directly minimize $\mathcal{L}_{CF}$, and learning of transformation functions is also guided by $\mathcal{L}_{CF}$, there is no error superimposition issue. Nevertheless, there is another challenge for joint-training methods – the *ineffective learning problem* – that is, because learning transformation functions is only guided by the recommendation error which is based on the final model output layer, the long distance from transformation layers to the model output layer from the perspective of backpropagation leads to ineffective model learning.

Moreover, a common issue with both separate-training and joint-training methods is the *unified transformation problem*. Concretely, almost all existing separate-training and joint-training methods adopt unified transformation functions (either a linear transformation or a neural network based non-linear transformation) for all users or items under the assumption that users (or items) keep the same relative relationships in both the auxiliary representation space and CF space. This assumption seldom holds because auxiliary information is usually noisy and complex, and cannot be directly aligned with the CF space. Thus, a unified transformation process is not effective and can generate low-quality CF representations.

Our contributions. To address these three challenges, we propose a new model **Heater**, which is designed to keep the advantages of both separate-training and joint-training methods, while overcoming the drawbacks identified above. To deal with the problem of error superimposition and the problem of ineffective learning, we combine the structures of separate-training and joint-training methods together as the basic framework of Heater. The main procedure of training is that Heater first transforms auxiliary representations to intermediate representations, then further refines intermediate representations to final CF representations to minimize recommendation error. Meanwhile, we also require the intermediate representations to be as close as possible to high-quality pretrained CF representations to improve the model effectiveness. To further address the ineffective learning problem, we propose a **Randomized Training** mechanism in which we randomly feed pretrained CF representations or intermediate representations alternatively to the refining component of Heater. By doing this, the effectiveness of model training is protected by the high-quality pretrained CF representations even when the intermediate representations are of poor quality. Last, we propose a **Mixture-of-Experts Transformation**, which adopts the Mixture-of-Expert [30] structure as transformation functions so that Heater can provide 'personalized' transformations for different users and items to tackle the unified transformation issue. Furthermore, unlike most existing methods [17, 28, 29, 31, 35], the proposed Heater can simultaneously recommend for both cold start users and items, rather than requiring separate user-based and item-based models. Last, we conduct extensive experiments on three real-world datasets to show the effectiveness of Heater over state-of-the-art alternatives, and the effectiveness of each proposed component.

2 PROPOSED METHOD

In this section, we formalize the cold start recommendation problem, then introduce the fundamental framework of Heater, the Randomized Training and Mixture-of-Experts Transformation mechanisms.

2.1 Cold Start Recommendation

Assume we have N_w warm users $\mathcal{U}_w = \{1, 2, \ldots, N_w\}$ and M_w warm items $\mathcal{I}_w = \{1, 2, \ldots, M_w\}$, each of which has at least one historical interaction record. We denote the set of all historical records as $O = \{(u, i)\}$, where u indexes one user, and i indexes one item. We also have N_c cold start users $\mathcal{U}_c = \{1, 2, \ldots, N_c\}$ and M_c cold start items $\mathcal{I}_c = \{1, 2, \ldots, M_c\}$, all of which have zero historical

interaction record. For these cold start users and items, there are three recommendation tasks:

Task 1: recommend warm items from $\mathcal{I}_w$ to cold users in $\mathcal{U}_c$;
Task 2: recommend cold items from $\mathcal{I}_c$ to warm users in $\mathcal{U}_w$;
Task 3: recommend cold items from $\mathcal{I}_c$ to cold users in $\mathcal{U}_c$.

Note that most previous works only consider the one-sided cold start situation [17, 28, 29, 31, 35], i.e., there are only cold start users or cold start items in the system. In this paper, we consider the more complex situation where there are cold start users and items simultaneously in the system. Furthermore, we assume we have access to *auxiliary information* such as an external user profile and item content information for both warm and cold users and items, denoted as $\mathbf{U} \in \mathbb{R}^{(N_w+N_c)\times E_u}$ and $\mathbf{I} \in \mathbb{R}^{(M_w+M_c)\times E_i}$, where E_u and E_i are the auxiliary representation dimensions for users and items.

2.2 Heater Framework

As we have discussed, there are two main categories of cold start recommendation approaches – separate-training and joint-training methods, which are determined by the relationship between CF and the transformation functions in the model. Both of these have obvious advantages and disadvantages: separate-training methods make full use of the user-item interaction data while suffering from the error superimposition problem; joint-training methods are free of the error superimposition issue but face the ineffective learning problem. Thus, to overcome these problems, we propose a framework to integrate the two distinct structures.

Due to the special characteristic that users or items involved during inference time are never seen before during training, the execution of Heater is different for training and inference. In the following, we first focus on the training process, then describe how to do inference by a trained model for cold start situations.

Training. Separate-training approaches learn CF representations and auxiliary-to-CF transformation separately by two independent losses $\mathcal{L}_{CF}$ and $\mathcal{L}_{trans}$, thus, increase of each of them leads to worse cold start recommendations (the error superimposition issue). To avoid this error superimposition issue, we adopt the joint-training structure as the foundation for Heater basic framework as shown in Figure 2. Generally, Heater takes one warm user auxiliary representation $\mathbf{U}_u \in \mathbb{R}^{1\times E_u}$ and one warm item auxiliary representation $\mathbf{I}_i \in \mathbb{R}^{1\times E_i}$ as inputs, and by multiple layers of transformation $\mathbf{U}_u$ and $\mathbf{I}_i$ are transformed to the CF representations $\mathbf{P}'_u$ and $\mathbf{Q}'_i$, of which the dot product is the predicted preference score $\hat{R}_{u,i} = (\mathbf{P}'_u)^T\mathbf{Q}'_i$ for the given user-item pair. More specifically, we first transform $\mathbf{U}_u$ and $\mathbf{I}_i$ to intermediate representations $\mathbf{U}'_u$ and $\mathbf{I}'_i$ by two transformation functions f_U and f_I as $\mathbf{U}'_u = f_U(\mathbf{U}_u)$ and $\mathbf{I}'_i = f_I(\mathbf{I}_i)$. Then, we further refine the representations by a multi-layer perceptron (MLP) (ϕ_U for user side, ϕ_I for item side) to get user and item CF representations: $\mathbf{P}'_u = \phi_U(\mathbf{U}'_u)$, $\mathbf{Q}'_i = \phi_I(\mathbf{I}'_i)$.

However, because the transformation layers f_U and f_I are trained based on the recommendation error $\mathcal{L}_{CF}$ which is calculated by the model output layer $\hat{R}$, the long distance from f_U and f_I to the output layer from the view of backpropagation makes it difficult to learn effective parameters for f_U and f_I, which can further impact the learning effectiveness of the whole model (the ineffective learning issue). Hence, to address this, we input high-quality user and item pretrained CF representations (denoted as $\mathbf{P}_u \in \mathbb{R}^{1\times D}$ and $\mathbf{Q}_i \in \mathbb{R}^{1\times D}$) from a pretrained CF model to help guide the learning processing for f_U and f_I by setting a *similarity constraint* to minimize the difference between the intermediate representations $\mathbf{U}'_u$ and $\mathbf{I}'_i$ and pretrained CF representations $\mathbf{P}_u$ and $\mathbf{Q}_i$:

$$\min \ \|\mathbf{U}'_u - \mathbf{P}_u\|_F^2 + \|\mathbf{I}'_i - \mathbf{Q}_i\|_F^2. \tag{1}$$

With the similarity constraint, f_U and f_I are guided by the final recommendation error as well as the high-quality pretrained CF representations, leading to more effective f_U and f_I.

Heater can be trained by any popular top-k recommendation loss function, such as Sum Squared Error (SSE) loss, cross-entropy loss, or Bayesian Personalized Ranking (BPR) loss [22]. In this paper, we use SSE loss because most of existing baselines [17, 28, 29, 31, 36] adopt this loss, and it shows good empirical performance. The objective function of Heater can be written as:

$$\min_{\Theta} \mathcal{L} = \sum_{(u,i)\in O\cup O^-} \|\hat{R}_{u,i} - R_{u,i}\|_F^2 + \frac{\alpha}{2}(\|\mathbf{U}'_u - \mathbf{P}_u\|_F^2 + \|\mathbf{I}'_i - \mathbf{Q}_i\|_F^2) + \frac{\lambda}{2}\|\Theta\|_F^2,$$

where O^- is the negative samples randomly generated based on O; $R_{u,i}$ is the ground-truth preference with value 1 if $(u,i) \in O$, 0 otherwise; α is the trade-off weight for similarity constraint; and λ is the trade-off weight for regularization.

Inference. With a trained Heater, it is straightforward to provide recommendations, which is similar to the training process. The only difference is that we do not need the similarity constraint and pretrained CF representations shown in Equation 1, and we only input the auxiliary representations of cold users and items into the model. For Task 1 mentioned in Section 2.1 – recommending warm items to cold users – assume we want to provide recommendation for the cold user u. All we need is to input all the $(\mathbf{U}_u, \mathbf{I}_i)$ pairs into Heater to calculate $\hat{R}_{u,i}$, where $i \in \mathcal{I}_w$, and show items with top scores in descending order to user u. In the same way, we can generate recommendations for Task 2 and Task 3.

The Heater framework can also be applied to cases where there is only auxiliary information for users or items (e.g., CiteULike and LastFM datasets introduced in Section 4.1), and the side (user side or item side) without auxiliary information is warm side (all users in the side or all items in the side are warm). During training, the only modifications needed are first removing the corresponding similarity constraint for the warm side, and then directly input the pretrained CF representations $\mathbf{P}_u$ (or $\mathbf{Q}_i$) as $\mathbf{U}'_u$ (or $\mathbf{I}'_i$) to calculate the final CF representations $\mathbf{P}'_u$ (or $\mathbf{Q}'_i$) for the warm side. During inference, similarly, we directly use the pretrained CF representations as input for the warm side.

2.3 Randomized Training

The introduced similarity constraint in the Heater framework (shown in Equation 1) is capable to guide the learning of transformation functions f_U and f_I, and thus improves the quality of $\mathbf{U}'$ and $\mathbf{I}'$ because the pretrained CF representations are known to be of high quality. As a result, the Heater framework should have higher model learning effectiveness than conventional joint-training methods.

However, even if we have the similarity constraint, there is always information loss between $\mathbf{P}$ (or $\mathbf{Q}$) and $\mathbf{U}'$ (or $\mathbf{I}'$) due to the low quality of auxiliary representations and the structural limitation of

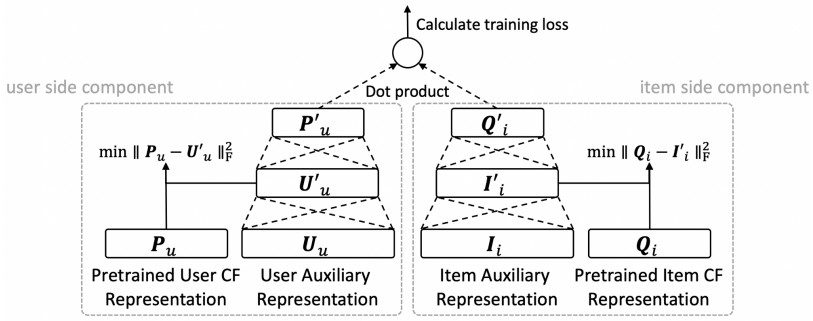


Figure 2: The framework of Heater, which incorporate structures of the separate-training and joint-training methods to solve the error superimposition and ineffective learning problems.

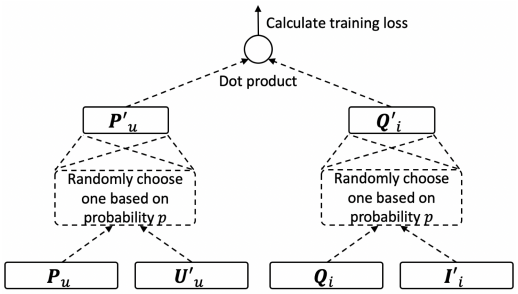


Figure 3: Randomized Training: during training, randomly feed pretrained CF representations or transformed auxiliary representations alternatively to generate final representations $\mathbf{P}'_u$ and $\mathbf{Q}'_i$.

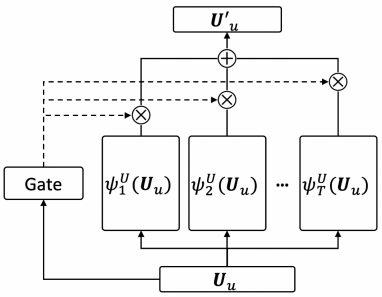


Figure 4: Mixture-of-Experts Transformation: apply T multi-layer perceptrons as experts to transform auxiliary representations, weighted sum outputs of experts as final output.

the transformation functions. This will lead to ineffective learning for ϕ_U and ϕ_I due to the low quality of $\mathbf{U}'$ and $\mathbf{I}'$, and further decrease the quality of final CF representations $\mathbf{P}'$ and $\mathbf{Q}'$. To address this, we propose a Randomized Training strategy, which does not only feed user and item transformed auxiliary representations $\mathbf{U}'$ and $\mathbf{I}'$ to generate final representations $\mathbf{P}'$ and $\mathbf{Q}'$ during training, but also uses the pretrained CF representations $\mathbf{P}$ and $\mathbf{Q}$ in a stochastic way as demonstrated in Figure 3. Note that the proposed Randomized Training is only for the training and does not influence the inference process introduced in Section 2.2.

Concretely, first, we need to pre-define a hyper-parameter $p \in [0, 1]$ representing the probability of using pretrained CF representations $\mathbf{P}$ and $\mathbf{Q}$ for training. Then, during the training process, for a given training sample (u, i), based on p, we randomly choose whether to use $\mathbf{U}'_u$ or $\mathbf{P}_u$ to generate $\mathbf{P}'_u$, and whether to use $\mathbf{I}'_i$ or $\mathbf{Q}_i$ to generate $\mathbf{Q}'_i$. Note that the random processes of user and item sides are independent, and they can even have different values of probability p.

By using both auxiliary representations and pretrained CF representations alternatively, we can alleviate the ineffective learning problem because the high-quality pretrained CF representations can help guide ϕ_U and ϕ_I to learn effective parameters, especially when $\mathbf{U}'$ and $\mathbf{I}'$ are not of high quality. The choice of p depends on the quality of the auxiliary information. If the auxiliary representations contain rich information about the users and items with limited noise, then we can use a small p, otherwise, we need a large p to ensure the effectiveness of the model.

2.4 Mixture-of-Experts Transformation

Last, we turn to address the unified transformation problem of existing methods. Recall that previous works apply the same transformation process f_U (or f_I) to map auxiliary representations into the CF space for all the users (or all items). In other words, they assume the relationships between users (or items) in the auxiliary representation space are the same as the relationships in the CF space. But this assumption can seldom hold because the auxiliary information is usually noisy and complex. For example, two users may have a large distance in the auxiliary representation space because they have little common information in their profiles, but they could have similar preference, i.e., small distance in CF space. A unified transformation function cannot handle this situation. Thus, an algorithm which is able to assign 'personalized' transformations to different users (or items) is required. As a result, we propose to adopt Mixture-of-Experts [30] to implement f_U and f_I for every single user and item.

The Mixture-of-Experts Transformation (as shown in Figure 4) consists of T experts, where each is a MLP, denoted as $\psi_1^U, \ldots, \psi_T^U$ for user side, $\psi_1^I, \ldots, \psi_T^I$ for item side. All of the experts take the same input $\mathbf{U}_u$ (or $\mathbf{I}_i$), and the final output of the Mixture-of-Experts Transformation is a weighted sum of the outputs of all experts. The formulation of the user side is:

$$MoE(\mathbf{U}_u) = g_1\psi_1^U(\mathbf{U}_u) + g_2\psi_2^U(\mathbf{U}_u) + \ldots + g_T\psi_T^U(\mathbf{U}_u), \quad (2)$$

where $g_1, g_2, \ldots, g_T$ are the weights for experts, which are calculated by another one-layer perceptron: $\mathbf{g} = \varphi(\mathbf{W}^T\mathbf{U}_u + \mathbf{b})$, where $\mathbf{g} \in \mathbb{R}^T$ is the vector consisted of $g_1, g_2, \ldots, g_T$; and $\varphi(\cdot)$ is the activation function, we use *tanh* in this work. The reason why we do

not use *softmax* for the weights is that experts here work in a collaborative way rather than an exclusive way, thus every expert should have an independent weight. And *tanh* empirically outperforms *softmax*. Note that the formulation for the item side is similar.

To better explain the effect of Mixture-of-Experts Transformation, we assume all experts are linear transformation matrices, denoted as $\mathbf{V}_1^U, \mathbf{V}_2^U, \ldots, \mathbf{V}_T^U$ for user side, $\mathbf{V}_1^I, \mathbf{V}_2^I, \ldots, \mathbf{V}_T^I$ for item side. Then, the output of Mixture-of-Experts Transformation for user u is:

$$MoE(\mathbf{U}_u) = (g_1\mathbf{V}_1^U + g_2\mathbf{V}_2^U + \ldots + g_T\mathbf{V}_T^U)^T\mathbf{U}_u, \tag{3}$$

where $(g_1\mathbf{V}_1^U + g_2\mathbf{V}_2^U + \ldots + g_T\mathbf{V}_T^U)$ generates a new transformation matrix specifically for user u based on her own auxiliary representation, which achieves our goal that assigning different transformations to different users (or items). By doing this, the auxiliary representations $\mathbf{U}$ and $\mathbf{I}$ can be transformed into the CF space more effectively than using a unified transformation function. It is also similar to the idea of applying meta-learning to solve cold start recommendation [35], which generates a unique logistic regression model for each user based on her historical interactions and then applies the logistic regression model on cold start items. Therefore, the Mixture-of-Experts Transformation can also be viewed as a meta-learning based method.

3 RELATED WORK

Warm Start Recommendation.
Recommender systems have been studied for many years, with an early emphasis on explicit rating based recommendation tasks. Various algorithms based on Collaborative Filtering have been proposed [4, 14, 18, 23–25]. However, since explicit ratings are not as widespread as implicit feedback (such as clicks or views), implicit top-k recommendation is receiving increasing attention [10, 11, 16, 20, 22]. Among these, Bayesian Personalized Ranking (BPR) [22] is one of the most influential ones, which adopts a pair-wise ranking based loss function. In this paper, we also focus on the implicit top-k recommendation problem, but aim to provide recommendations for cold start users and/or items, which is impossible for the conventional models mentioned above.

Recently, many neural networks based recommendation algorithms are proposed [6, 9, 27, 38–40], which show more promising performance than traditional models due to the non-linearity and structural complexity of neural networks. Among these, one breakthrough is Neural Collaborative Filtering (NCF) proposed by He et al. [9], which generalizes matrix factorization by neural networks and adopts multiple hidden layers and non-linear activation functions, resulting in significant performance improvement. Inspired by these pioneers, the proposed model in this work is also based on neural networks and utilizes the neural structure of matrix factorization introduced in NCF.

Cold Start Recommendation.
One inherent drawback of the recommendation algorithms mentioned above is that they can not recommend for users or items without any historical interaction data. However, this cold start scenario is common in many real-world applications. As discussed in Section 1, there are two main categories of modern approaches: separate-training and joint-training methods.

Separate-training methods separate the learning of the CF model and the transformation from auxiliary representations to CF representations. One typical group of methods is to first pretrain a CF model on the user-item interaction data and generate pretrained CF representations, then learn the transformation functions based on the pretrained CF representations. One example is LinMap proposed by Gantner et al. [7], which learns a linear transformation matrix to map auxiliary representations to pretrained CF representations. Another example is DeepMusic [33], which use deep neural networks to transform music audio signals to pretrained CF representations. Another group of methods [3, 26, 31] combines the learning of CF and transformation functions into one model, but they have their own loss components in the objective function and a trade off of learning strength between them must be achieved. For example, CMF proposed by Singh et al. [31] learns a matrix factorization and a linear transformation function in one model, whose core idea is similar to LinMap.

Joint-training methods learn CF and transformation functions jointly to minimize the recommendation error on the user-item interaction data in one model. A typical example is DropoutNet proposed by Volkovs et al. [36], which transforms auxiliary representations to CF representations by multi-layer perceptions, and learns the model parameters by minimizing a recommendation error. Another thread of methods directly learns a transformation from auxiliary representations to preference scores [17, 28] without generating any intermediate CF representations, which is inspired by Autoencoder based CF [27].

Beside these two directions, there are some works adapting the idea of meta-learning to the cold start recommendation field [15, 35]. Vartak ei al. [35] propose to generate different models for different users based on their historical preferences, then apply these user-specific models to auxiliary representations of cold start tweets to predict user preference towards tweets. Lee et al. [15] build a recommender that is able to adapt to users or items with very few interaction records (instead of no feedback as in our work) so that it can perform well on new users or items. Because [15] is not the same cold start setting as we discuss in this paper, we do not compare with it.

4 EXPERIMENTS

In this section, we empirically evaluate the proposed model over three cold start recommendation tasks and three datasets from different domains. We aim to answer four key research questions: **RQ1** How does Heater perform compared with state-of-the-art cold start recommendation models? **RQ2** How effective are the proposed similarity constraint, Randomized Training, and Mixture-of-Experts Transformation mechanisms? **RQ3** What are the impact of three key hyper-parameters: similarity constraint weight α, Randomized Training probability p, and number of experts T in Mixture-of-Experts Transformation? **RQ4** What is the impact of the quality of pretrained CF representations on Heater compared with other models that also take pretrained representations as input?

4.1 Datasets

Datasets for evaluating cold start recommendation need rich auxiliary information for users and items, thus we use three real-world

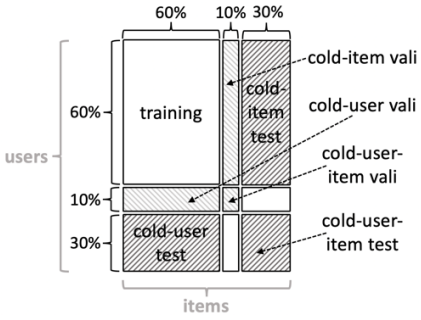


Figure 5: Validation and test set splitting for XING dataset.

datasets commonly used to evaluate the three different cold start recommendation tasks introduced in Section 2.1:

CiteULike [37] is a dataset recording user preferences towards scientific articles. There are 5,551 users, 16,980 articles and 204,986 user-like-article interactions in the dataset. Besides, we have the abstracts of the articles as the auxiliary information for the item side, but there is no auxiliary information for the user side, hence we *evaluate Task 2 on CiteULike*. Following the processing of [36], we first generate an 8,000 dimension feature vector by calculating tf-idf of top 8,000 words for each item, and then keep the top 300 dimensions after dimensionality reduction by SVD. As a result, we have a $16,980 \times 300$ item auxiliary representation $\mathbf{I}$.

LastFM [5] consists of 1,892 users, 17,632 music artists as items to be recommended, and *92,834 user-listen-to-artist interactions* (rather than the user-tag-artist interactions as used in previous work [28], because user-listen-to-artist interaction is more general and the data is sparser). In this dataset, we have the social relationships between all the users, thus we have a $1,892 \times 1,892$ user auxiliary representation $\mathbf{U}$, but have no auxiliary information for the item side. Therefore, we *evaluate Task 1 on LastFM*.

XING [2] is a subset of the ACM RecSys 2017 Challenge dataset, which contains 106,881 users, 20,519 jobs as items to be recommended to users, and 4,306,183 user-view-job interactions. We have user profile information such as current job, location and education level. For items, we have career level, tags, and other related information. Following the processing of [36], we have a $106,881 \times 831$ user auxiliary representation $\mathbf{U}$ and a $20,519 \times 2,738$ item auxiliary representation $\mathbf{I}$. We *evaluate all three tasks on XING*.

For CiteULike, we use the same training and test splitting of [36], but further select 30% of items and all records of them from the test set in [36] as validation set and the remaining part as our test set. For LastFM, we randomly select 10% of users and all their records as the validation set and 30% of items and all their records as the test set. For XING, as shown in Figure 5, we select cold start users and cold start items randomly for validation set and test set in the same way, and generate cold-user, cold-item, and cold-user-item validation sets and test sets (6 sets in total). The detailed statistics of the datasets are shown in Table 1.

4.2 Experimental Setup

Metrics. We adopt three different ranking evaluation metrics to evaluate model performance: *Precision@k* (*P@k*), *Recall@k* (*R@k*) and *NDCG@k*. The detailed definitions of these metrics can be found in [9, 19]. We report $k = \{20, 50, 100\}$ in this paper.

Baselines. We consider eight state-of-the-art cold start recommendation algorithms to compare with the proposed model:

KNN [29] generates recommendations by conventional nearest neighbor algorithm. The user-user or item-item similarity is computed by the given auxiliary representations. This method works for Task 1 and 2 but not Task 3.

CMF [31] combines matrix factorization and auxiliary representations to CF representations transformation together into one objective function and trains these two parts simultaneously. CMF works for Task 1 and Task 2 but not Task 3.

LinMap [7] inputs pretrained CF representations and learns a matrix to transform auxiliary representations to pretrained CF representations. LinMap can work for all three tasks.

NLinMap is similar to [33], which applies deep neural networks to extract features from auxiliary representation to transform the auxiliary representations to CF space. We use a MLP of architecture $300 \rightarrow 300 \rightarrow 200$ for CiteULike, a MLP of architecture $800 \rightarrow 400 \rightarrow 200$ for LastFM and XING. All hidden layers have *ReLU* as the activation function. NLinMap can work for all three tasks.

LoCo [28] is a linear low-rank regression method, which learns a low-rank transformation matrix to directly transform auxiliary representations to final predicted preference scores. It can only work for Task 1 and 2 but not Task 3.

LWA [35] is a meta-learning based algorithm which constructs different logistic regression classifiers for different users based on their historical records. The user-specific logistic regression takes the auxiliary representation of one cold start item as input and predicts whether the user will like input item or not. LWA can only work for Task 2.

DropoutNet [36] inputs both pretrained CF representations and auxiliary representations into a MLP and randomly dropouts pretrained CF representations during training. It works for all tasks.

LLAE [17] applies the idea of zero-shot learning to solve cold start recommendation problems. Similar to LoCo, LLAE also learns a transformation matrix to directly transform auxiliary representation to predicted preference scores.

Reproducibility. Code, data, and experimental settings are at https://github.com/Zziwei/Heater–Cold-Start-Recommendation. We implement the proposed model by Tensorflow [1] and Adam [12] optimizer. For the hyper-parameters, we fix the CF latent factor dimension as 200, and set the learning rate as 0.005, the mini-batch size as 1024 for all models. Besides, we re-sample negative samples in each epoch and set the negative sampling rate 5 for all models. Then we tune other hyper-parameters by grid search on validation sets. More specifically, for Heater, we set the regularization weight $\lambda = 0.0001$ for CiteULike and XING, and $\lambda = 0.001$ for LastFM. We set the similarity constraint weight $\alpha = 0.0001$, set the Randomized Training probability $p = 0.5$, set the number of experts in Mixture-of-Experts Transformation as 5, have one hidden layer activated by *tanh* of dimension 200 as the expert , and have one hidden layer of dimension 200 activated by *tanh* as ϕ_U and ϕ_I for all 3 datasets.

Heater and some of the baselines require pretrained CF representations as input. Hence, we train a Bayesian Personalized Ranking (BPR) [22] model with latent factors of 200 dimensions, *L2* regularization weight 0.001, and learning rate as 0.005 for the three datasets, and use the learned latent factors of BPR as $\mathbf{P}$ and $\mathbf{Q}$.

	Training				Validation			Test		
	#user	#item	#record	density	#user	#item	#record	#user	#item	#record
CiteULike	5,551	13,584	164,210	0.22%	5,551	1,018	13,037	5,551	2,378	27,739
LastFM	1,136	12,850	55,810	0.38%	189	12,850	9,209	567	12,850	27,815
XING-U	64,129	12,312	1,549,242	0.20%	10,688	12,312	258,497	32,064	12,312	775,837
XING-I	64,129	12,312	1,549,242	0.20%	64,129	2,051	275,782	64,129	6,156	756,638
XING-UI	64,129	12,312	1,549,242	0.20%	10,688	2,051	45,807	32,064	6,156	379,730

Table 1: Statistics of training, validation and test sets in the three datasets. XING-U represents XING dataset with cold start users (for Task 1), XING-I represents XING with cold start items (for Task 2), and XING-UI represents XING with both cold statr users and items (for Task 3).

All experiments are performed on a desktop machine with 32GB memory, an 8 core Intel i7-4820k 3.7GHz CPU and an Nvidia GeForce GTX Titan X GPU with 12 GB memory. The runtime of one epoch for Heater is 7s for CiteUlike, 6s for LastFM, and 4 minutes and 58 seconds for XING. Heater can converge within 100 epochs.

4.3 RQ1: Heater vs. Baselines

We begin by comparing the performance of Heater with eight state-of-the-art alternatives on three datasets. *Recall@k*, *Precision@k* and *NDCG@k* ($k = \{20, 50, 100\}$) are shown in Table 2. XING-U represents recommending warm items to cold start users (Task 1) in XING dataset. Similarly, XING-I represents recommending cold start items to warm users (Task 2), and XING-UI represents recommending cold start items to cold start users (Task 3). The best baselines are marked in bold, and the relative improvement (denoted as Δ) of Heater over the best baselines are also calculated. As we can see from the table, for both metrics and all datasets, Heater is able to outperform other models for different cold start recommendation tasks. We also calculate the p-value of paired t-test for the relative improvement rates, showing the improvements are statistically significant. Note that LWA cannot work for situations with cold start users, and there is no sufficient memory to run LLAE on XING. Hence, we do not report results of LWA for LastFM, XING-U, and XING-UI, and do not report results of LLAE for all three XING cases. Besides, KNN, CMF, LoCo, LWA and LLAE cannot work for Task 3, thus we do not report their results for XING-UI.

In addition to the outstanding cold start recommendation performance, another advantage of Heater is that it is able to address all three cold start tasks simultaneously by one time of training. Unlike LinMap and NLinMap, which have to first solve Task 1 and Task 2 one by one, then generate cold start recommendations for Task 3 based on the trained models from Task 1 and Task 2.

4.4 RQ2: Ablation Study

Next, we turn to investigate the effects of different components of Heater. We compare the complete version of Heater with three variations: (i) Heater without the similarity constraint (noted as w/o SC), which puts no constraint on $\mathbf{U}'_u$ and on $\mathbf{I}'_i$; (ii) Heater without Mixture-of-Experts Transformation (noted as w/o MoET), which just adopts a linear transformation as the transformation functions f_U and f_I; and (iii) Heater without Randomized Training (noted as w/o RT), which adopts the Heater basic framework as introduced in Section 2.2 with Mixture-of-Experts Transformation. Table 3 shows *Recall@20*, *Precision@20*, and *NDCG@20* of the three models over all three datasets. Generally, Heater outperforms all variations for all metrics and all datasets, which indicates that the proposed similarity constraint, Mixture-of-Experts Transformation and Randomized Training mechanisms are effective and help to improve the cold start recommendation quality. Another observation is that for different datasets and different cold start recommendation tasks, the performance improvement brought by the similarity constraint, Mixture-of-Experts Transformation or Randomized Training are different. For instance, for CiteULike, Randomized Training improves the *NDCG@20* by only 1.61% but Mixture-of-Experts Transformation improves by 10.32% and similarity constraint improves by 8.55%, however, for LastFM, Mixture-of-Experts Transformation improves only by 0.43% while Randomized Training improves by 4.90% and similarity constraint improves by 55.22%.

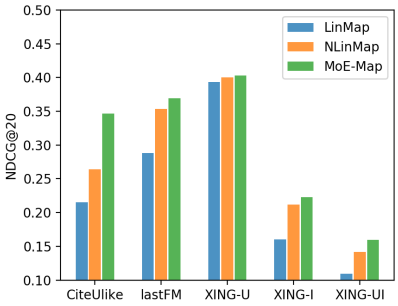


Figure 6: *NDCG@20* comparison between MoE-Map model, LinMap and NLinMap.

Moreover, to further study the effectiveness of Mixture-of-Experts Transformation, we also implement a variation of NLinMap with the original MLP replaced by a Mixture-of-Expert Transformation (with the same structure as it in Heater described in section 4.2). The variation is denoted as MoE-Map. Comparisons of *NDCG@20* between MoE-Map and LinMap and NLinMap are present in Figure 6. From the figure we can observe that MoE-Map outperforms LinMap and NLinMap for all cases. As a result, we can conclude that Mixture-of-Experts Transformation is effective for auxiliary representation transformation.

4.5 RQ3: Impact of Hyper-parameters

Next, we study the impact of three hyper-parameters: the similarity constraint weight α, the Randomized Training probability p, and the number of experts in Mixture-of-Experts Transformation T.

Similarity constraint weight α.

We first vary the similarity constraint weight α in $\{0, 5e-6, 1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4, 1e-3\}$ and set the other hyper-parameters as the same as described in Section 4.2. α controls the strength of similarity constraint and is directly connected with

		CiteULike (Task 2)			LastFM (Task 1)			XING-U (Task 1)			XING-I (Task 2)			XING-UI (Task 3)		
		@20	@50	@100	@20	@50	@100	@20	@50	@100	@20	@50	@100	@20	@50	@100
KNN	Recall	.2192	.3853	.5208	.1354	.2343	.3406	.1222	.2191	.3011	.0732	.1054	.1421	-	-	-
	Precision	.0479	.0359	.0254	.3065	.2108	.1473	.1498	.1070	.0739	.0408	.0244	.0167	-	-	-
	NDCG	.1500	.2310	.2905	.3537	.2956	.3625	.1722	.2154	.2581	.0740	.0906	.1061	-	-	-
LinMap	Recall	.2351	.4197	.5738	.1152	.2039	.2946	.2902	.4475	.5556	.1631	.2988	.4355	.1118	.2180	.3314
	Precision	.0586	.0420	.0297	.2500	.1776	.1289	.3502	.2179	.1348	.0983	.0725	.0528	.0675	.0531	.0405
	NDCG	.2150	.3049	.3743	.2880	.2547	.3152	.3933	.4558	.4103	.1605	.2296	.2865	.1095	.1635	.2115
CMF	Recall	.2663	.4450	.5774	.1319	.2203	.2979	.2472	.3934	.4901	.0735	.1948	.3257	-	-	-
	Precision	.0642	.0448	.0299	.2880	.1922	.1302	.3003	.1933	.1198	.0433	.0472	.0396	-	-	-
	NDCG	.2289	.3219	.3822	.3332	.2833	.3347	.3488	.4082	.4571	.0628	.1250	.1802	-	-	-
LoCo	Recall	.2918	.5001	.6411	.1374	.2346	.3205	.2464	.3946	.5071	.2070	.3624	.4827	-	-	-
	Precision	.0695	.0492	.0327	.3054	.2079	.1421	.3009	.1939	.1240	.1263	.0866	.0581	-	-	-
	NDCG	.2503	.3543	.4189	.3586	.3071	.3646	.3538	.4128	.4698	.2230	.2984	.3495	-	-	-
NLinMap	Recall	.2748	.4614	.6251	.1398	**.2468**	**.3462**	**.2966**	**.4496**	.5533	.2114	.3665	.5003	.1410	.2551	.3699
	Precision	.0686	.0469	.0324	.3065	**.2156**	**.1520**	**.3579**	**.2191**	.1344	.1268	.0879	.0603	.0859	.0620	.0449
	NDCG	.2641	.3585	.4295	.3535	**.3105**	**.3771**	**.4001**	**.4583**	**.5147**	.2118	.2891	.3455	.1418	.1992	.2469
LWA	Recall	.3223	.4966	.6264	-	-	-	-	-	-	.1976	.3446	.4813	-	-	-
	Precision	.0773	.0505	.0328	-	-	-	-	-	-	.1196	.0829	.0579	-	-	-
	NDCG	.2960	.3917	.4524	-	-	-	-	-	-	.2008	.2750	.3316	-	-	-
DropoutNet	Recall	.3275	.5092	**.6518**	.1351	.2371	.3384	.2417	.4219	**.5638**	**.2215**	**.3714**	**.5091**	**.1436**	**.2599**	**.3780**
	Precision	.0770	.0500	.0331	.3001	.2097	.1496	.2921	.2058	**.1371**	**.1334**	**.0900**	**.0619**	**.0879**	**.0637**	**.0464**
	NDCG	.3089	.4026	.4674	.3439	.3012	.3687	.2761	.3920	.4646	**.2236**	**.3007**	**.3589**	**.1454**	**.2052**	**.2553**
LLAE	Recall	**.3622**	**.5313**	.6434	**.1403**	.2342	.3221	-	-	-	-	-	-	-	-	-
	Precision	**.0841**	**.0535**	**.0335**	**.3107**	.2072	.1421	-	-	-	-	-	-	-	-	-
	NDCG	**.3249**	**.4217**	**.4758**	**.3658**	.3092	.3674	-	-	-	-	-	-	-	-	-
Heater	Recall	**.3727**	**.5533**	**.6852**	**.1451**	**.2575**	**.3686**	**.3074**	**.4727**	**.5810**	**.2420**	**.3984**	**.5368**	**.1609**	**.2895**	**.4113**
	Precision	**.0894**	**.0552**	**.0352**	**.3221**	**.2279**	**.1620**	**.3714**	**.2308**	**.1413**	**.1431**	**.0956**	**.0648**	**.0973**	**.0699**	**.0498**
	NDCG	**.3731**	**.4673**	**.5278**	**.3705**	**.3270**	**.3994**	**.4150**	**.4798**	**.5345**	**.2372**	**.3171**	**.3759**	**.1566**	**.2211**	**.2720**
Δ	Recall	2.9%*	4.1%**	5.1%**	3.4%**	4.3%**	6.5%**	3.6%**	5.1%**	3.1%**	9.3%**	7.3%**	5.4%**	12.0%**	11.4%**	8.8%**
	Precision	6.3%**	3.2%**	5.1%**	3.7%**	5.7%**	6.6%**	3.8%**	5.3%**	3.1%**	7.3%**	6.2%**	4.7%**	10.7%**	9.7%**	7.3%**
	NDCG	14.8%**	10.8%**	10.9%**	1.3%*	5.3%**	5.9%**	3.7%**	4.7%**	3.8%**	6.1%**	5.5%**	4.7%**	7.7%**	7.7%**	6.5%**

Table 2: *Recall@k*, *Precision@k*, **and** *NDCG@k* **of all baselines and Heater. '-' represents unavailable result: KNN, CMF, LoCo, LWA and LLAE cannot work for Task 3, thus there is no result for them on XING-UI; LWA cannot work for Task 1 thus there is no result for LWA on LastFM and XING-U; LLAE run into out-of-memory error on XING dataset thus there is no result of LLAE on XING-U and XING-I. ** indicates that the relative improvement rates are statistically significant for $p < 0.01$, * for $p < 0.05$ judged by paired t-test.**

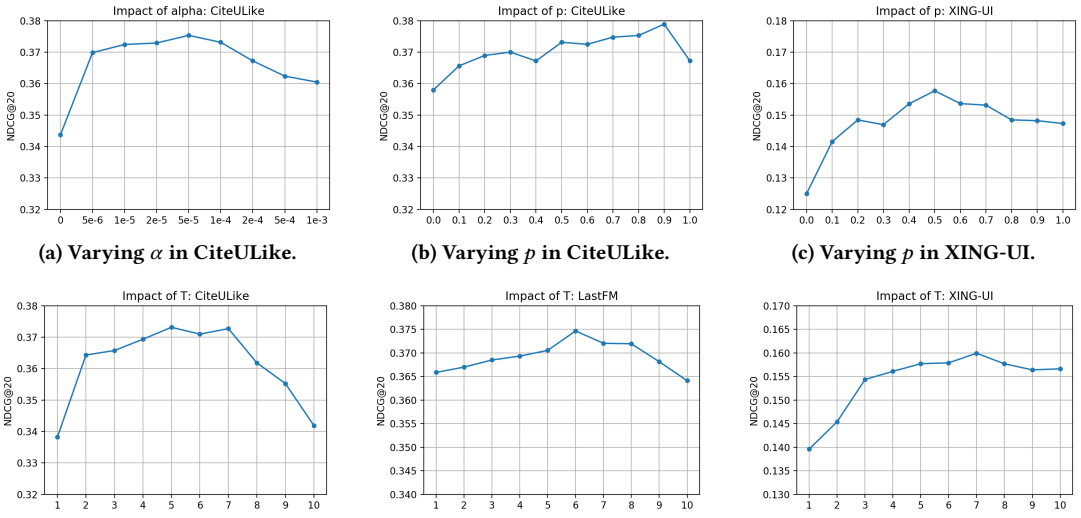
		CiteULike	LastFM	XING-U	XING-I	XING-UI
Heater	*R@20*	**.3727**	**.1451**	**.3074**	**.2420**	**.1609**
	P@20	**.0894**	**.3221**	**.3714**	**.1431**	**.0973**
	NDCG@20	**.3731**	**.3705**	**.4150**	**.2372**	**.1566**
w/o SC	*R@20*	.3273	.0944	.2723	.2092	.1252
	P@20	.0818	.2112	.3307	.1259	.0781
	NDCG@20	.3437	.2387	.3595	.2053	.1263
w/o MoET	*R@20*	.3406	.1431	.2856	.2160	.1454
	P@20	.0827	.3185	.3449	.1291	.0890
	NDCG@20	.3382	.3689	.3753	.2132	.1434
w/o RT	*R@20*	.3654	.1415	.2407	.1843	.1534
	P@20	.0887	.3095	.2946	.1099	.0929
	NDCG@20	.3672	.3532	.3145	.1833	.1511

Table 3: *Recall*@20, *Precision*@20 **and** *NDCG*@20 **of proposed Heater, Heater w/o similarity constraint, Heater w/o Mixture-of-Experts Transformation, and Heater w/o Randomized Training. MoET represents Mixture-of-Experts Transformation, RT represents Randomized Training.**

model effectiveness. For conciseness, we only report results on CiteULike because the patterns on other datasets show similar results as CiteULike. *NDCG*@20 results of Heater on CiteULike are shown in Figure 7a, where we can observe that with α increasing, the cold start recommendation quality first improves and then decreases. This is reasonable because small α causes underfitting for the auxiliary representation transformation, while large α does not only give rise to overfitting for the transformation, but also decreases the ratio of parameters updating due to recommendation loss. For CiteULike, the best performance is achieved when $\alpha = 5e-5$, and for other datasets the best performances are achieved around the same value.

Randomized Training probability p.

Then, we vary the Randomized Training probability p in the range of $[0.0, 1.0]$ with step 0.1. Our goal is to know how p influences the randomized training process and how to set a reasonable p to maximize the effect of Randomized Training. Because experiments on LastFM, XING-U and XING-I show similar patterns as XING-UI, thus we only plot *NDCG*@20 results of CiteULike in Figure 7b and XING-UI in Figure 7c. Generally, with p growing from 0.0 to 1.0, *NDCG*@100 first increases then decreases, but the peak performances are different for different datasets: the best metrics are achieved when $p = 0.9$ in CiteULike and $p = 0.5$ in XING-UI



(a) Varying α in CiteULike. **(b) Varying p in CiteULike.** **(c) Varying p in XING-UI.**

(d) Varying T in CiteULike. **(e) Varying T in LastFM.** **(f) Varying T in XING-UI.**

Figure 7: *NDCG@20* results of Heater with different hyper-parameters.

(around 0.5 for other datasets). We can draw two conclusions: (i) Randomized Training is effective for improving cold start recommendation: when p is within a reasonable range, the performance improves with p increasing; and (ii) large p is not always helpful because f_U and f_I are trained less in this case.

Number of experts in Mixture-of-Experts Transformation T. Last, we investigate the impact of the number of experts T in the Mixture-of-Experts Transformation. T controls the model complexity of the transformation component, and theoretically larger T (within reasonable range) is supposed to produce better performance. We experiment with T varying from 1 to 10 with step 1, and the empirical results on three datasets are shown in Figure 7d, 7e and 7f (XING-U, XING-I and XING-UI show similar patterns, thus we only show results of XING-UI here). Generally, on all datasets, with T increasing from 1, the performance improves first and arrives at a peak. However, the best T for different datasets are different, which is reasonable because the auxiliary representations in different datasets have distinct characteristics and requires different degrees of complexity for Mixture-of-Experts Transformation. The best choice of T is 5 for CiteULike, 6 for LastFM, and 7 for XING-UI.

Another direction of changing the complexity of the Mixture-of-Experts Transformation is to increase the number of layers in each expert. But adding more layers dramatically increases the computational cost, and does not bring much performance improvement, hence we do not show experimental results here. Besides, we only use one layer for one expert, thus, we cannot change the model complexity by changing the hidden layer dimension because it must be the same as the dimension of pretrained CF representations.

4.6 RQ4: Impact of Pretrained CF Quality.

We last study the impact of the quality of pretrained CF representations on Heater compared with other models which also take pretrained CF representations as input. We want to know with better pretrained CF representations, does Heater perform better? And is Heater relatively robust to the change of pretrained CF model compared with other models? For conciseness, we only show results on CiteULike. Experiments on other datasets show similar patterns. We generate user and item CF representations by training BPR on user-item interactions of warm start users and items, and we fix all hyper-parameters of BPR except the number of training epochs to generate representations of different qualities. Because we can only use the training set to train and evaluate BPR, thus we further split the training set shown in Table 1 into 90% for training, and 10% for evaluating. Then, with different quality of pretrained representations, we compare the cold start recommendation performance of DropoutNet, LinMap, NLinMap, and Heater. *NDCG@20* results on CiteULike are shown in Figure 8, where we involve five sets of pretrained representations of different qualities, x-axis represents the *NDCG@20* of BPR trained by different numbers of epochs (6, 8, 12, 16 and 30 epochs respectively, and the epoch number of experiment in Section 4.3 is 30), y-axis represents the cold start recommendation performances of models. From this we can observe that for all pretrained representations of different qualities, Heater always outperforms alternatives, and as the quality of BPR increases, the performance of all models improves. Moreover, the performance of Heater is much more consistent when the quality

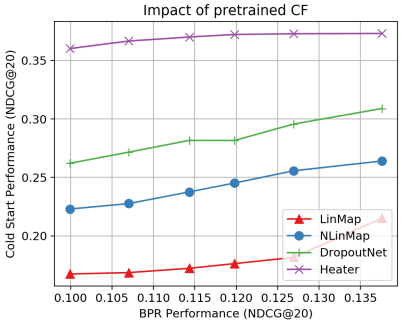


Figure 8: *NDCG@20* results on CiteULike of LinMap, NLinMap, DropoutNet and Heater with different pretrained CF representations of varying quality.

of BPR representations differs, while other models have a larger range of performance changes. Note that *NDCG*@20 of BPR is lower than that of cold start recommendation performance because they are under different experiment setups: BPR is evaluated by recommending 13, 584 items to 5, 551 users, cold start models are evaluated by recommending 2, 378 items to 5, 551 users.

5 CONCLUSIONS

In this work, we propose a novel model called Heater to address the cold start recommendation problem. There are three innovative features of Heater, which are designed to address three key challenges in existing cold start recommendation algorithms: i) a combined framework incorporating the structures of separate-training and joint-training methods to avoid the error superimposition issue and improve the model effectiveness; ii) the Randomized Training strategy to further promote quality of model learning; and iii) the Mixture-of-Experts Transformation mechanism to provide 'personalized' transformations for users and items. Empirical results on three public datasets show the performance improvement of Heater over state-of-the-art alternatives. In the future, we plan to investigate how to apply some of the ideas of Heater to solve the data sparsity problem, i.e., improve the recommendation performance on users or items with limited historical interaction data.

ACKNOWLEDGMENTS

This work is, in part, supported by NSF #IIS-1841138.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16.

[2] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. 2017. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 372–373.

[3] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 91–98.

[4] Daniel Billsus and Michael J Pazzani. 1998. Learning Collaborative Information Filters.. In *Icml*, Vol. 98. 46–54.

[5] Ivan Cantador, Peter L Brusilovsky, and Tsvi Kuflik. 2011. Second workshop on information heterogeneity and fusion in recommender systems. (2011).

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[7] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations.. In *ICDM*, Vol. 10. Citeseer, 176–185.

[8] James Loke Hale. 2019. More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute.

[9] 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*. International World Wide Web Conferences Steering Committee, 173–182.

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[11] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).

[13] Michal Kompan and Mária Bieliková. 2010. Content-based news recommendation. In *International conference on electronic commerce and web technologies*. Springer.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[15] 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*. ACM, 1073–1082.

[16] Gai Li and Weihua Ou. 2016. Pairwise probabilistic matrix factorization for implicit feedback collaborative filtering. *Neurocomputing* 204 (2016), 17–25.

[17] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From Zero-Shot Learning to Cold-Start Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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

[19] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural Personalized Ranking for Image Recommendation. In *Proceedings of 11th ACM International Conference on Web Search and Data Mining*.

[20] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[21] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[23] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. *Application of dimensionality reduction in recommender system-a case study*. Technical Report. Minnesota Univ Minneapolis Dept of Computer Science.

[24] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*. Citeseer, 27–28.

[26] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.

[27] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. 2017. Low-rank linear cold-start recommendation from social data. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[29] Suvash Sedhain, Scott Sanner, Darius Braziunas, Lexing Xie, and Jordan Christensen. 2014. Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 345–348.

[30] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[31] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

[32] Kit Smith. 2019. 53 Incredible Facebook Statistics and Facts.

[33] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[34] Robin Van Meteren and Maarten Van Someren. 2000. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 47–56.

[35] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *Advances in neural information processing systems*. 6904–6914.

[36] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*. 4957–4966.

[37] 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*. ACM, 448–456.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *arXiv preprint arXiv:1905.08108* (2019).

[39] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM.

[40] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.