Appendix

A. Discussion on Representation Power and Expressiveness

We compare IDCF with other related models from two perspectives in order to shed more lights on the advantages and differences of our model.

Comparison of Representation Power. We first provide a comparison with related works on methodological level as a clear elaboration for essential differences of our work to others. In Fig. 5, we present an intuitive comparison with general CF model, local-graph-based inductive CF and item-based CF model. As shown in Fig. 5(a), general collaborative filtering assumes user-specific embeddings for users and learn them collaboratively among all the users in one dataset. It disables inductive learning due to such learnable embeddings. Item-based model, as shown in Fig. 5(b), leverages embeddings of user's historically rated items to compute user's embeddings via some pooling methods. The learnable parameters only lie in the item space. It suffers from limited representation capacity compared to general CF model that assumes both user and item embeddings. Besides, local-graph-based inductive model (e.g. (Zhang & Chen, 2020)), shown in Fig. 5(b), extracts local subgraph structures within 1-hop neighbors of each user-item pair (i.e., rated items of the user and users who rated the item) from a bipartite graph of all the observed user-item ratings and use GNNs to encode such graph structures for rating prediction. Note that the model requires that the local subgraphs do not contain user and item indices, so it cannot output user-specific embeddings. Its representation power is limited since it cannot represent diverse user preferences with arbitrary rating history on items. Also it cannot output user representations. Differently, our model IDCF, as shown in Fig. 5(d) adopts item-based embedding as initial states for query users to compute attention scores on key users and aggregate the embeddings (i.e., meta latents) of key users to estimate user-specific embeddings for query users, which maintains ability to produce user representations with enough representation power and meanwhile achieves inductive learning.

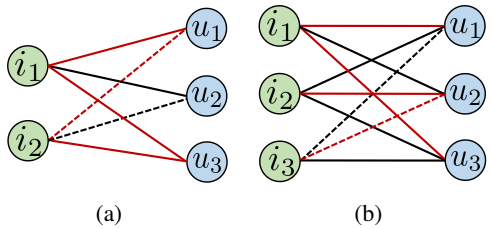


Figure 4. Comparison of expressiveness. Feature-driven and local-graph-based models fail in (a) and (b), respectively. IDCF works effectively in both cases with superior expressiveness.

Comparison of Expressiveness. We provide a comparison with feature-driven and local-graph-based inductive (matrix completion) models through two cases in Fig. 4 so as to highlight the superior expressiveness of IDCF. Here we assume ratings are within $\{-1, 1\}$ (positive, denoted by red line, and negative, denoted by black line). The solid lines are observed ratings for training and dash lines are test ratings. In Fig. 4(a), we consider test ratings (u_1, i_2) and (u_2, i_2). For local-graph-based models, the 1-hop local subgraphs of (u_1, i_2) (resp. (u_2, i_2)) consists of $\{i_1, i_2, u_1, u_3\}$ (resp. $\{i_1, i_2, u_2, u_3\}$) and the subgraph structures are different for two cases due to the positive (resp. negative) edge for (u_1, i_1) (resp. (u_2, i_1)). The local-graph-based models can give right prediction for two ratings relying on different structures. Also, CF models and IDCF can work smoothly in this case, relying on different rating history of u_1 and u_2. However, feature-driven models will fail once u_1 and u_2 have the same features though two users have different rating history. In Fig. 4(b), we consider test ratings (u_1, i_3) and (u_2, i_3). The 1-hop local subgraphs of (u_1, i_3) (resp. (u_2, i_3)) consists of $(i_1, i_2, i_3, u_1, u_3)$ (resp. $(i_1, i_2, i_3, u_2, u_3)$) and the subgraph structures are the same. Thus, the local-graph-based models will fail to distinguish two inputs and give the same prediction for (u_1, i_3) and (u_2, i_3). Differently, CF models and IDCF can recognize that u_3 has similar rating patterns with u_1 and different from u_2, thus pushing the embedding of u_1 (resp. u_2) close to (resp. distant from) u_3, which guides the model to right prediction. Note that the first case becomes a common issue when the feature space is small while the second case becomes general when the rating patterns of users are not distinct enough throughout a dataset, which induces similar local subgraph structures. Therefore, IDCF enjoys superior expressiveness for input data than feature-driven and local-graph-based inductive (matrix completion) models. Also, it maintains as good expressiveness as (transductive) CF models.

B. Proofs in Section 3

B.1. Proof of Theorem 1

Proof. The proof is trivial by construction. Assume the optimal $\mathbf{P}_2$ for Eq. (3) as $\mathbf{P}_2^*$. Since $\mathbf{P}_1$ given by Eq. (1) is column-full-rank, for any column vector $\mathbf{p}_{u'}^*$ in $\mathbf{P}_2^*$ ($u' \in \mathcal{U}_2$), there exists $\mathbf{c}_{u'}^*$ such that ${\mathbf{c}_{u'}^*}^\top \mathbf{P}_1 = \mathbf{p}_{u'}^*$. Hence, $\mathbf{C}^* = [\mathbf{c}_{u'}^*]_{u' \in \mathcal{U}_2}$ is a solution for Eq. (2) and gives $\mathcal{D}_{\mathcal{S}_q}(\hat{R}_2, R_2) < \epsilon$. □

B.2. Proof of Theorem 2

Proof. With fixed a true rating matrix R_2 to be learned and a probability distribution $\mathcal{P}$ over $[M_q] \times [N]$, which is unknown to the learner, we consider the problem under the framework of standard PAC learning. We can treat the

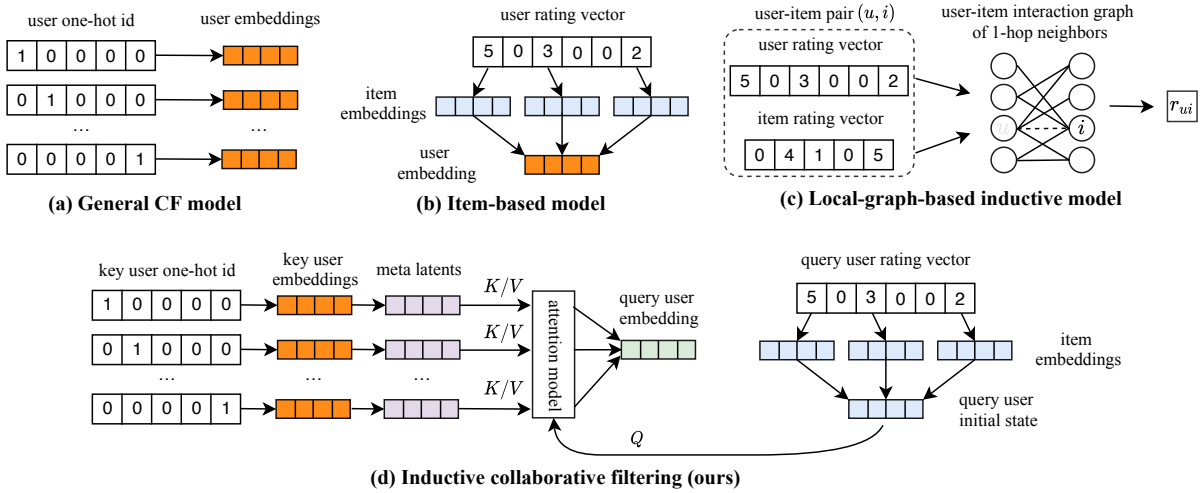


Figure 5. Comparison with related works on methodological level. We highlight the advantage of proposed model IDCF: it can 1) inductively compute user representations (or embeddings) and meanwhile 2) maintain enough capacity as general CF models.

matrix R_2 as a function $(u', i) \to r_{u'i}$. Let $\mathcal{R}$, a set of matrices in $\mathbb{R}^{M_q \times N}$, denotes the hypothesis class of this problem. Then the input to the learner is a sample of R_2 denoted as

$$\mathcal{T} = \left((u'_t, i_t, r_{u'_t i_t}) | (u'_t, i_t) \in \mathcal{S}_q\right),$$

where $\mathcal{S}_q = \{(u'_t, i_t)\} \in ([M_q] \times [N])^{T_2}$ is a set with size T_2 containing indices of the observed entries in R_2 and each (u', i) in $\mathcal{S}_q$ is independently chosen according to the distribution $\mathcal{P}$. When using $\mathcal{T}$ as training examples for the learner, it minimizes the error $\mathcal{D}_{\mathcal{S}_q}(\hat{R}_2, R_2) = \frac{1}{T_2}\sum_{(u',i)\in\mathcal{S}_q} l(r_{u'i}, \hat{r}_{u'i})$. We are interested in the generalization error of the learner, which is defined as

$$\mathcal{D}(\hat{R}_2, R_2) = \mathbb{E}_{(u',i)\in\mathcal{P}}[l(r_{u'i}, \hat{r}_{u'i})].$$

The (empirical) Rademacher complexity of $\mathcal{R}$ w.r.t. the sample $\mathcal{T}$ is defined as

$$Rad_{\mathcal{T}}(\mathcal{R}) = \frac{1}{T_2}\mathbb{E}_\sigma\left[\sup_{\hat{R}_2\in\mathcal{R}} \sum_{t=1}^{T_2} \sigma_t \hat{r}_{u'_t i_t}\right],$$

where $\sigma_t \in \{-1, 1\}$ is a random variable with probability $Pr(\sigma_t = 1) = Pr(\sigma_t = -1) = \frac{1}{2}$. Assume $l(\cdot, \cdot)$ is L-Lipschitz w.r.t. the first argument and $|l(\cdot, \cdot)|$ is bounded by a constant. Then a general result for generalization bound of $\mathcal{R}$ is

Lemma 1. *(Generalization bound (Mohri et al., 2012)): For a sample $\mathcal{T}$ with random choice of $\mathcal{S}_q = ([M_q]\times[N])^{T_2}$, it holds that for any $\hat{R}_2 \in \mathcal{R}$ and confidence parameter $0 < \delta < 1$,*

$$Pr(\mathcal{D}(\hat{R}_2, R_2) \le \mathcal{D}_{\mathcal{S}_q}(\hat{R}_2, R_2) + G) \ge 1 - \delta, \quad (14)$$

where,

$$G = 2L \cdot Rad(\mathcal{R}) + O\left(\sqrt{\frac{\ln(1/\delta)}{T_2}}\right).$$

Based on the lemma, we need to further estimate the Rademacher complexity in our model to complete the proof. In our model, $\hat{R}_2 = \mathbf{C}^\top \mathbf{P}_k \mathbf{Q}$ and the entry $\hat{r}_{u'i}$ is given by $\hat{r}_{u'i} = \mathbf{p}_{u'}^\top \mathbf{q}_i = \mathbf{c}_{u'}^\top \mathbf{P}_k \mathbf{q}_i$ (where $\mathbf{c}_{u'}$ is the u'-th colunm vector of $\mathbf{C}$). Define $\mathcal{C}$ as a set of matrices,

$$\mathcal{C} = \{\mathbf{A} \in [0,1]^{M_q \times M_k} : \|\mathbf{a}_{u'}\|_1 = \sum_{u=1}^{M_k} |a_{u'u}| = 1\}.$$

Then we have

$$\begin{aligned}
T_2 \cdot Rad_{\mathcal{T}}(\mathcal{R}) &= \mathbb{E}_\sigma\left[\sup_{\mathbf{C}\in\mathcal{C}} \sum_{t=1}^{T_2} \sigma_t \mathbf{c}_{u'_t}^\top \mathbf{P}_k \mathbf{q}_{i_t}\right] \\
&= \mathbb{E}_\sigma\left[\sup_{\mathbf{C}\in\mathcal{C}} \sum_{u'=1}^{M_q} \mathbf{c}_{u'}^\top \cdot \left(\sum_{t:u_t=u'} \sigma_t \hat{R}_{k,*i_t}\right)\right] \\
&\quad \text{(since } \hat{R}_{k,*i_t} = \mathbf{P}_k \mathbf{q}_{i_t}\text{)} \\
&\le H \cdot \mathbb{E}_\sigma\left[\sup_{\mathbf{A}\in\mathcal{A}} \sum_{u'=1}^{M_q} \mathbf{a}_{u'}^\top \cdot \left(\sum_{t:u_t=u'} \sigma_t \hat{R}_{k,*i_t}\right)\right] \\
&= H \cdot \mathbb{E}_\sigma\left[\sum_{u'=1}^{M_q} \max_{u\in[M_k]} \left(\sum_{t:u_t=u'} \sigma_t \hat{r}_{ui_t}\right)\right].
\end{aligned} \quad (15)$$

The last equation is due to the fact that $\mathbf{a}_{u'}$ is a probability distribution for choosing entries in $R_{k,*i_t}$, the i_t-th column of matrix $\hat{R}_k$. In fact, we can treat the $\max_{u\in[M_k]}$ inside

the sum over all $u' \in \mathcal{U}_2$ as a mapping κ from $u' \in [M_q]$ to $u \in [M_k]$. Let $\mathcal{K} = \{\kappa : [M_q] \to [M_k]\}$ be the set of all mappings from $[M_q]$ to $[M_k]$, and then the above formula can be written as

$$\mathbb{E}_\sigma \left[\sum_{u'=1}^{M_q} \max_{u \in [M_k]} \left(\sum_{t:u_t=u'} \sigma_t \hat{r}_{ui_t} \right) \right] \tag{16}$$

$$= \mathbb{E}_\sigma \left[\sup_{\kappa \in \mathcal{K}} \sum_{u'=1}^{M_q} \sum_{t:u_t=u'} \sigma_t \hat{r}_{\kappa(u'),i_t} \right] \tag{17}$$

$$= \mathbb{E}_\sigma \left[\sup_{\kappa \in \mathcal{K}} \sum_{t=1}^{T_2} \sigma_t \hat{r}_{\kappa(u_t),i_t} \right] \tag{18}$$

$$\leq B\sqrt{T_2} \cdot \sqrt{2M_q \log M_k}. \tag{19}$$

The last inequality is according to the Massart Lemma. Hence, we have

$$Rad_{\mathcal{T}}(\mathcal{R}) \leq HB\sqrt{\frac{2M_q \log M_k}{T_2}}. \tag{20}$$

Incorporating Eq. (20) into Eq. (14), we will arrive at the result in this theorem. □

C. Extensions of IDCF

IDCF can be extended to feature-based setting and deal with extreme cold-start recommendation where test users have no historical ratings. Here, we provide details of feature-based IDCF (IDCF-HY) which indeed is a hybrid model that considers both user features and one-hot user indices. Furthermore, we discuss in the views of transfer-learning and meta-learning that can be leveraged to enhance our framework as future study.

C.1. Hybrid Model with Features (IDCF-HY)

Assume $\mathbf{a}_u$ denotes user u's raw feature vector, i.e., a concatenation of all the features (often including binary, categorical and continuous variables) where categorical features can be denoted by one-hot or multi-hot vectors. If one has m user features in total, then $\mathbf{a}_u$ can be

$$\mathbf{a}_u = [\mathbf{a}_{u1}||\mathbf{a}_{u2}||\mathbf{a}_{u3}||\cdots||\mathbf{a}_{um}].$$

Then we consider user-sharing embedding function $\mathbf{y}_i()$ which can embed each feature vector into a d-dimensional embedding vector:

$$\mathbf{y}_u = [\mathbf{y}_1(\mathbf{a}_{u1})||\mathbf{y}_2(\mathbf{a}_{u2})||\mathbf{y}_3(\mathbf{a}_{u3})||\cdots||\mathbf{y}_m(\mathbf{a}_{um})].$$

Similarly, for item feature $\mathbf{b}_i = [\mathbf{b}_{i1}||\mathbf{b}_{i2}||\mathbf{b}_{i3}||\cdots||\mathbf{b}_{in}]$, we have its embedding representation:

$$\mathbf{z}_i = [\mathbf{z}_1(\mathbf{b}_{i1})||\mathbf{z}_2(\mathbf{b}_{i2})||\mathbf{z}_3(\mathbf{b}_{i3})||\cdots||\mathbf{z}_n(\mathbf{b}_{in})].$$

Also, we assume user-specific index embedding $\mathbf{p}_u$ and item-specific index embedding $\mathbf{q}_i$ for user u and item i, respectively, as is in Section 3. The prediction for user u's rating on item i can be

$$\hat{r}_{ui} = g_\theta(\mathbf{p}_u, \mathbf{y}_u, \mathbf{q}_i, \mathbf{z}_i), \tag{21}$$

where g_θ can be a shallow neural network with parameters denoted by θ. To keep notation clean, we denote $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m\}$ and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$. Then for key users in $\mathcal{U}_k$ with rating matrix R_k, we consider the optimization problem,

$$\min_{\mathbf{P}_k, \mathbf{Q}, \mathbf{Y}, \mathbf{Z}, \theta} \mathcal{D}_{\mathcal{S}_k}(\hat{R}_k, R_k), \tag{22}$$

based on which we get learned feature embedding functions $\mathbf{Y}$, $\mathbf{Z}$ as well as transductive embedding matrices $\mathbf{P}_k$, $\mathbf{Q}$ which we further use to compute inductive embeddings for query users.

For query users, feature embeddings can be obtained by the learned $\mathbf{Y}$ and $\mathbf{Z}$ in Eq. (22), i.e., $\mathbf{y}_{u'} = [\mathbf{y}_{u'1}(\mathbf{a}_{u'1})||\cdots||\mathbf{y}_{u'm}(\mathbf{a}_{u'm})]$ where $\mathbf{a}_{u'}$ is raw feature vector of user u'. Then we have a relation learning model h_w that consists of a multi-head attention function and use user feature as input $\mathbf{d}_{u'} = \mathbf{y}_{u'}$. The inductive user-specific representation can be given by $\mathbf{p}_{u'} = h_w(\mathbf{d}_{u'})$ (i.e., Eq. (5) and Eq. (6)), similar as the CF setting in Section 3. The rating of user u' on item i can be predicted by $\hat{r}_{u'i} = g_\theta(\mathbf{p}_{u'}, \mathbf{y}_{u'}, \mathbf{q}_i, \mathbf{z}_i)$. Also, the optimization for inductive relation model is

$$\min_{w,\theta} \mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q). \tag{23}$$

C.2. Extreme Cold-Start Recommendation

For cold-start recommendation where test users have no historical rating, we have no information about users if without any side information. In such case, most CF models would fail for personalized recommendation and degrade to a trivial one which outputs the same result (or the same distribution) to all the users using the popularity of items. For IDCF, the set $\mathcal{I}_{u'}$ in Eq. (4) would be empty for users with no historical rating, in which situation we can randomly select a group of key users to construct $\mathcal{I}_{u'}$ used for computing attentive scores with key users. Another method is to directly use average embeddings of all the key users as estimated embeddings for query users. In such case, the model degrades to ItemPop (using the numbers of users who rated the item for prediction).

On the other hand, if side information (such as user profile features) is available, our hybrid model IDCF-HY can leverage user features for computing inductive embeddings, which enables extreme cold-start recommendation. We apply this method to cold-start recommendation on Movielens-1M using features and the results are given in Appendix E.

C.3. Transfer Learning & Meta-Learning

Another extension of IDCF is to consider transfer learning on cross-domain recommendation tasks or when treating recommendation for different users as different tasks like (Lee et al., 2019). Transfer learning and meta learning have shown power in learning generalizable models that can adapt to new tasks. In our framework, we can also take advantage of transfer learning (few-shot learning or zero-shot learning) or mete-learning algorithms to train our relation learning model h_w. For example, if using model-agnostic meta-learning algorithm for the second-stage optimization, we can first compute one-step (or multi-step) gradient update independently for each user (or a group of clustering users) in a batch and then average them as one global update for the model. The meta-learning can be applied over different groups of users or cross-domain datasets.

D. Details in Implementations

We provide implementation details that are not presented in Section 5 in order for reproducibility.

D.1. Hyper-parameter Settings

We present details for hyper-parameter settings in different datasets. We use $L = 4$ attention heads for our inductive relation learning model among all the datasets. For Douban and ML-100K, each attention head randomly samples 200 key users for computing attention weights. For ML-1M, we set sample size as 500; for Amazon-Books and Amazon-Beauty, we set it as 2000. We use Adam optimizer and learning rates are searched within [0.1, 0.01, 0.001, 0.0001]. For pretraining, we consdier L2 regularization for user and item embeddings. The regularization weights are searched within [0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2]. The mini-batch sizes are searched within [64, 256, 512, 1024, 2048] to keep a proper balance between training efficiency and performance. For adaption stage, regularization weight λ is set as 10 for five datasets. Besides, different hyper-parameters for architectures are used in three implementations.

IDCF-NN. For Douban and ML-100K, we use embedding dimension $d = 16$ and neural size $48 - 32 - 32 - 1$ for f_θ. For ML-1M, ML-10M and Amazon-Books, we use $d = 32$ and neural size $96 - 64 - 64 - 1$ for f_θ.

IDCF-GC. For Douban and ML-100K, we use embedding dimension $d = 32$ and neural size $128 - 32 - 32 - 1$ for f_θ. For ML-1M, ML-10M and Amazon-Books, we use $d = 64$ and neural size $256 - 64 - 64 - 1$ for f_θ.

IDCF-HY. We use embedding size $d = 32$ for each feature in ML-1M as well as user-specific and item-specific index embeddings. The neural size of g_θ is set as $320-64-64-1$.

D.2. Evaluation Metrics

We provide details for our adopted evaluation metrics. In our experiments, we follow evaluation protocols commonly used in previous works in different settings. Three metrics used in our paper are as follows.

- **RMSE:** Root Mean Square Error is a commonly used metric for explicit feedback data and measures the averaged L2 distance between predicted ratings and ground-truth ratings:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathcal{I}^+} (\hat{r}_{ui} - r_{ui})^2}{|\mathcal{I}^+|}}. \tag{24}$$

- **AUC:** Area Under the ROC Curve is a metric for implicit feedback data. It measures general consistency between a ranking list of predicted scores and ground-truth ranking with 1's before 0's. More specifically, AUC counts the average area under the curve of true-positive v.s. false-positive curve:

$$AUC = \frac{\sum_{(u,i)\in\mathcal{I}^+} \sum_{(u',i')\in\mathcal{I}^-} \delta(\hat{r}_{u,i} > \hat{r}_{u',i'})}{|\mathcal{I}^+||\mathcal{I}^-|}, \tag{25}$$

  where $\mathcal{I}^+ = \{(u,i)|r_{ui} > 0\}$ and $\mathcal{I}^- = \{(u',i')|r_{u'j'} = 0\}$ denote the sets of observed user-item interaction pairs and unobserved user-item pairs respectively. The indicator $\delta(\hat{r}_{ui} > \hat{r}_{uj})$ returns 1 when $\hat{r}_{ui} > \hat{r}_{uj}$ and 0 otherwise. Since we only have ground-truth positive examples (clicked items) for users, we negatively sample five items as negative examples (non-clicked items) for each user-item rating in dataset, which composes the set $\mathcal{I}^-$.

- **NDCG:** Normalized discounted cumulative gain (Normalized DCG) measures the usefulness, or gain, of a recommended item based on its position in the result list. NDCG can be used to evaluate the model on both explicit and implicit feedback data. The gain is accumulated from the top of the recommendation list to the bottom, with the gain of each result discounted at lower ranks. The NDCG metric is computed for each user and can measures the averaged performance of personalized recommendation. Given the ranking list of recommended items to a user u, denoted as $\hat{\mathcal{K}}_u$, its DCG is defined as:

$$DCG = \sum_{i\in\hat{\mathcal{K}}_u} \frac{rel_i}{\log_2(i+1)}. \tag{26}$$

  where rel_i is the graded relevance of item i with user u. For explicit feedbacks, rel_i is the ground-truth rating of user u on item i. For implicit feedbacks, $rel_i = 1$ for

observed user-item interaction and $rel_i = 0$ otherwise. The normalized discounted cumulative gain, or NDCG, is computed as:

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{27}$$

where $IDCG@K$ is ideal discounted cumulative gain: $IDCG@K = \sum_{i \in \mathcal{K}_u} \frac{rel_i}{\log_2(i+1)}$, and $\mathcal{K}_u$ represents the ground-truth ranking list of relevant items (ordered by their ground-truth ratings/interactions by user u).

E. More Experiment Results

E.1. Impacts of different splits for key and query users

In our experiments in Section 5, we basically consider users with more than δ training ratings as $\mathcal{U}_1$ and the remaining as $\mathcal{U}_2$, based on which we construct key users and query users to study model's performance on few-shot query users (for inductive interpolation) and new unseen users (for inductive extrapolation). Here we provide a further discussions on two spliting ways and study the impact on model performance.

- **Threshold**: we select users with more than δ training ratings as $\mathcal{U}_1$ and users with less than δ training ratings as $\mathcal{U}_2$.
- **Random**: we set a ratio $\gamma \in (0, 1)$ and randomly sample $\gamma \times 100\%$ of users in the dataset as $\mathcal{U}_1$. The remaining users are grouped as $\mathcal{U}_2$.

We consider $\delta = [20, 30, 40, 50, 60, 70]$ and $\gamma = [0.97, 0.85, 0.75, 0.68, 0.62, 0.57]$ (which exactly gives the same ratio of $|\mathcal{U}_1|$ and $|\mathcal{U}_2|$ as corresponding δ in threshold split[5]) in Movielens-1M dataset. For each spliting, we also consider two situations for key users and query users: 1) inductive learning for interpolation on few-shot query users, i.e., the first-starge training is on the training ratings of key users $\mathcal{U}_k = \mathcal{U}_1$ and the second-stage training is on the training ratings of query users $\mathcal{U}_k = \mathcal{U}_2$; inductive learning for extrapolation on zero-shot new users, i.e., the two-stage trainings are both on the training ratings of a same group of users $\mathcal{U}_k = \mathcal{U}_q = \mathcal{U}_1$. We test the model on the testing ratings of users in $\mathcal{U}_2$. The results of IDCF-NN are presented in Table 5 where we report test RMSEs on all the users, few-shot query users and zero-shot new users.

As we can see from Table 5, with *threshold split*, as δ increases (we have fewer key users and more query users and they both have more training ratings on average), test RMSEs for query users exhibit a decrease. The reason is two-folds: 1) since key usres have more training ratings, the transductive model can learn better representations; 2) since query users have more training ratings, the inductive model would have better generalization ability. On the other hand, with different spliting thresholds, test RMSEs for new users remain in a fixed level. The results demonstrate that the performance of IDCF on new unseen users is not sensitive to different splitting thresholds. However, with *random split*, when γ decreases (also we have fewer key users and more query users but their average training ratings stay unchanged), RMSEs for new users suffer from an obvious decrease. One possible reason is that when we use smaller ratio of key users with random split, the 'informative' key users in the dataset are more likely to be ignored. (Recall that, as is shown in Fig. 3(c), there exist some important key users that give high attention weights on query users.) If such key users are missing, the performance would be affected due to insufficient expressive power of the inductive representation model.

Comparing threshold split with random split, we can find that when using the same ratio of key users and query users (i.e., the same column in Table 5), RMSEs on new users with threshold split are always better than those with random split. Such observation again shows that key users with more historical ratings would be more informative for providing useful information to inductive representation learning on query users, and again echo the results in Fig. 3(d) which demonstrates that important key users who give large attention weights on query users tend to exist in users with sufficient historical ratings.

E.2. Ablation Studies

In Table 6 we present the results of ablation study on ML-1M and Amazon-Books datasets. We compare IDCF-GC with 1) RD-Item (using randomized item embeddings), 2) Trans-User (directly optimizing $\mathbf{P}_q$ in Eqn. (2)) and 3) Meta-Path (using meta-path *user-item-user* in the observed user-item bipartite graph to determine users' neighbors for message passing). The results show that RD-Item performs much worse than IDCF-GC since the randomized item embeddings may provide wrong signals for both graph learning and final prediction. Compared with Trans-User, IDCF-GC significantly outperforms it over a large margin. The reason is that directly optimizing $\mathbf{P}_q$ would lead to serious overfitting since query users have few training data. Furthermore, we can see that Meta-Path provides inferior performance than IDCF-GC in Table 1. The reason is that meta-path can only identify limited relations from observed bipartite graph that often has missing/noisy links, while IDCF learns and explores useful semantic relations for sufficient message passing.

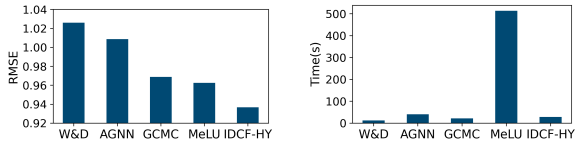
[5] For example, using $\gamma = 0.97$ in random split will result in the same sizes of $\mathcal{U}_1$ and $\mathcal{U}_2$ as using $\delta = 20$ in threshold split.

Table 5. Test RMSEs on all the users (All), few-shot query users (FS) and new users (New) of IDCF-NN in Movielens-1M using different splits for key and query users. (Lower RMSE is better)

Threshold	δ	20	30	40	50	60	70
	All (RMSE)	0.8440	0.8437	0.8439	0.8440	0.8444	0.8451
	FS (RMSE)	0.9785	0.9525	0.9213	0.9166	0.9202	0.9160
	New (RMSE)	0.9945	0.9942	0.9902	0.9883	0.9911	0.9929
Random	γ	0.97	0.85	0.75	0.68	0.62	0.57
	All (RMSE)	0.8446	0.8536	0.8587	0.8637	0.8669	0.8689
	FS (RMSE)	0.8863	0.8848	0.8760	0.8805	0.8824	0.8855
	New (RMSE)	0.9901	0.9923	0.9956	0.1001	1.0198	1.0262

Table 6. Ablation studies on ML-1M and Amazon-Books datasets. (Lower RMSE and Higher AUC/NDCG are better)

Method	**ML-1M**				**Amazon-Books**			
	Query		**New**		**Query**		**New**	
	RMSE	**NDCG**	**RMSE**	**NDCG**	**AUC**	**NDCG**	**AUC**	**NDCG**
IDCF-GC	**0.944**	**0.940**	**0.957**	**0.942**	**0.938**	**0.946**	**0.921**	**0.930**
RD-Item	1.014	0.843	1.023	0.835	0.665	0.701	0.832	0.821
Trans-User	0.992	0.876	-	-	0.845	0.821	-	-
Meta-Path	0.959	0.912	0.981	0.892	0.910	0.916	0.882	0.901



(a) Test RMSEs (b) Training time per epoch

Figure 6. Performance comparison for extreme cold-start recommendation on ML-1M with user profile features.

E.3. Cold-Start with User Features

We also wonder if our inductive model can handle extreme cold-start users with no historical rating [6]. Note that cold-start users are different and more challenging compared to new (unseen) users. For new users, the model can still use historical ratings as input features during inference, though it cannot be trained on these ratings. To enable cold-start recommendation, we leverage user attribute features in Movielens-1M. We use the dataset provided by (Lee et al., 2019), which contains attribute features and split warm-start and cold-start users. For IDCF, we also adopt the training algorithm of inductive learning for extrapolation and treat the warm-start users as key and query users. We user the *warm-start* users' training ratings for model training and the *cold-start* users' test ratings for test. We compare with Wide&Deep network (Cheng et al., 2016), GCMC (using feature vectors) and two recently proposed methods for cold-start recommendation: graph-based model AGNN (Qian et al., 2019) and meta-learning model MeLU (Lee et al., 2019).

[6]In some literature, cold-start users also mean users with few historical ratings (for training or/and inference). Here we consider extreme cold-start recommendation for users with no historical rating for both training and inference.

Fig. 6(a) gives the test RMSEs for all the models. It shows that our IDCF-HY outperforms the competitors, achieving 2.6% improvement of RMSE over the best one MeLU even on the difficult zero-shot recommendation task. The result indicates that IDCF is a promising approach to handle new users with no historical behavior in real-world dynamic systems. We also compare the training time per epoch of each method in Fig. 6(b). By contrast, IDCF-HY is much faster than MeLU and as efficient as AGNN and GCMC.