Federated Recommendation with Additive Personalization

Zhiwei Li
School of Computer Science
University of Technology Sydney
15 Broadway, Sydney, NSW 2007
lizhw.cs@outlook.com

Guodong Long
School of Computer Science
University of Technology Sydney
15 Broadway, Sydney, NSW 2007
guodong.long@uts.edu.au

Tianyi Zhou
University of Maryland, College Park
College Park, MD, USA
zhou@umiacs.umd.edu

Abstract

Building recommendation systems via federated learning (FL) is a new emerging challenge for next-generation Internet service and privacy protection. Existing FL models share item embedding across clients while keeping the user embedding private and local on the client side. However, identical item embedding cannot capture users' individual differences in perceiving the same item and may lead to poor personalization. Moreover, dense item embedding in FL results in expensive communication costs and latency. To address these challenges, we propose **Fed**erated **R**ecommendation with **A**dditive **P**ersonalization (**FedRAP**), which learns a global view of items via FL and a personalized view locally on each user. FedRAP encourages a sparse global view to save FL's communication cost and enforces the two views to be complementary via two regularizers. We propose an effective curriculum to learn the local and global views progressively with increasing regularization weights. To produce recommendations for a user, FedRAP adds the two views together to obtain a personalized item embedding. FedRAP achieves the best performance in FL setting on multiple benchmarks. It outperforms recent federated recommendation methods and several ablation study baselines.

1 Introduction

Recommendation systems have emerged as an important tool and product to allocate new items a user is likely to be interested in and they deeply change daily lives. These systems typically rely on central servers to aggregate user data, digital activities, and preferences in order to train a model to make accurate recommendations [1]. However, uploading users' personal data, which often contains sensitive privacy information, to central servers can expose them to significant privacy and security risks [2]. Furthermore, recent government regulations on privacy protection [3] necessitate that user data need to be stored locally on their devices, rather than being uploaded to a global server.

As a potential solution to the above problem, federated learning (FL) [4] enforces data localization and trains a globally shared model in a distributed manner, to be specific, by alternating between two fundamental operations, i.e., client-side local model training and server-side aggregation of local models. It achieves great success in some applications such as Google keyboard query suggestions [5–7]. However, the heterogeneity across clients, e.g., non-identical data distributions, can significantly slow down the FL convergence, resulting in client drift or poor global model performance on individual clients. A variety of non-IID FL methods [8–10] is developed to address the challenge by finding a better trade-off between global consensus and local personalization in model training.

Preprint. Under review.

However, most of them mainly focus on similar-type classification tasks, which naturally share many common features, and few approaches are designed for recommendation systems, which have severer heterogeneity among users, focus more on personalization, but suffer more from local data deficiency.

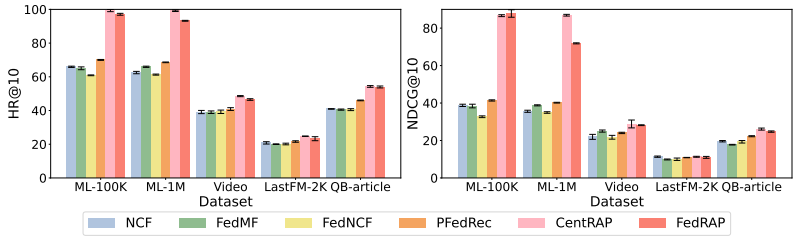


Figure 1: Evaluation metrics (%) on five real-world recommendation datasets. *CentRAP* is a centralized version (upper bound) of *FedRAP*. *FedRAP* outperforms all the FL methods by a large margin.

Federated Recommendation. In the quest to foster knowledge exchange among heterogeneous clients without breaching user privacy, scholars are delving into federated recommendation systems. This has given rise to the burgeoning field of Federated Recommendation Systems (FRSs) [11]. FRSs deal with client data originating from a single user, thereby shaping the user's profile [12]. The user profile and rating data stay locally on the client side while the server is allowed to store candidate items' information. To adhere to FL constraints while preserving user privacy, FRSs need to strike a fine balance between communication costs and model precision to yield optimal recommendations. Several recent approaches [12, 9, 13] have emerged to confront these hurdles. Most of them share a partial model design [14, 15] in which the item embedding is globally shared and trained by the existing FL pipeline while the user function/embedding is trained and stays local. However, they ignore the heterogeneity among users in perceiving the same item, which is item-specific and cannot be captured by personalized user embedding. Moreover, FL of item embedding requires expensive communication of a dense matrix between clients and a server, especially for users interested in many items.

Main Contributions. To address the challenges of heterogeneity and personalization in FRSs, we propose **Fed**erated **R**ecommendation with **A**dditive **P**ersonalization (**FedRAP**), which balances global knowledge sharing and local personalization by applying an additive model to item embedding and reduces communication cost/latency with sparsity. FedRAP follows horizontal FL assumption [16] with distinct user embedding and unique user datasets but shared items. Unlike previous methods, as depicted in Fig. 2, FedRAP integrates a bipartite personalization: personalized and private user embedding, and item additive personalization via summation of a user-specific item embedding matrix $\mathbf{D}^{(i)}$ and a globally shared item embedding matrix $\mathbf{C}$ updated via server-side FL aggregations. Two regularizers are applied: one encouraging the **sparsity of $\mathbf{C}$ (to reduce the communication cost/overhead)** and the other enforcing difference between $\mathbf{C}$ and $\mathbf{D}^{(i)}$ (to be complementary). In earlier training, additive personalization may hamper performance due to the time-variance and overlapping between $\mathbf{C}$ and $\mathbf{D}^{(i)}$; to mitigate this, regularization weights are incrementally increased by **a curriculum transitioning from full to additive personalization**. Therefore, FedRAP is able to utilize locally stored partial ratings to predict user ratings for unrated items by **considering both the global view and the user-specific view of the items.** In experiments on four real-data benchmarks, FedRAP significantly outperforms SOTA FRSs approaches.

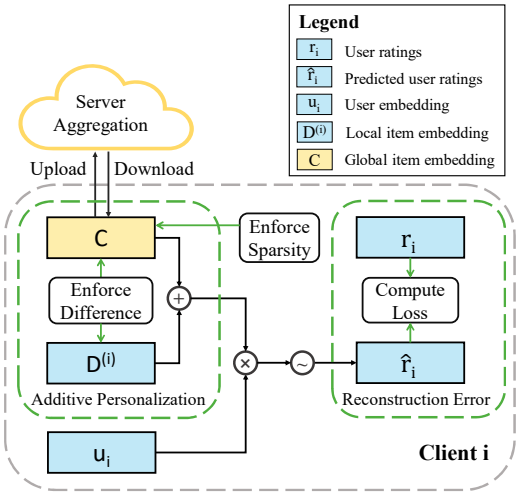


Figure 2: **The framework of FedRAP**. For each user-i, FedRAP utilizes its ratings r_i as labels to locally train a user embedding $\mathbf{u}_i$ and a local item embedding $\mathbf{D}^{(i)}$. By adding $\mathbf{D}^{(i)}$ to the global item embedding $\mathbf{C}$ from the server, i.e., **additive personalization**, FedRAP creates a personalized item embedding based on both shared knowledge and personal perspective and thus produces better recommendations. Since clients and server only communicate $\mathbf{C}$, FedRAP enforce its sparsity to reduce the communication cost and encourage its difference to $\mathbf{D}^{(i)}$ so they are complementary.

2 Related Work

Personalized Federated Learning. To mitigate the effects of heterogeneous and non-IID data, numerous works have been proposed [6, 8, 17–23]. In this paper, we concentrate on works based on personalized federated learning (PFL). Unlike the works that aim to learn a global model, PFL seeks to train individualized models for distinct clients [24], often necessitating server-based model aggregation [25–27]. Several studies [28, 19, 26] accomplish federated personalized learning by introducing various regularization terms between local and global models. To address federated cold-start recommendations, [29] proposes an integrated approach, where clients provide user features, an item server supplies item features, and a server orchestrates federated training. Meanwhile, [21] and [30] focus on personalized model learning, with the former promoting the closeness of local models via variance metrics and the latter enhancing this by clustering users into groups and selecting representative users for training, instead of random selection.

Federated Recommendation Systems. FRSs are unique in that they typically have a single user per client [31], with privacy-protective recommendations garnering significant research attention. Several FRS models leverage stochastic gradient updates with implicit feedback [28] or explicit feedback [12, 13, 32, 30]. FedMF [2] introduces the first federated matrix factorization algorithm based on non-negative matrix factorization [33], albeit with limitations in privacy and heterogeneity. Meanwhile, PFedRec [34] offers a bipartite personalization mechanism for personalized recommendations, but neglects shared item information, potentially degrading performance. Although existing works have demonstrated promising results, they typically only consider user-side personalizations, overlooking diverse emphases on the same items by different users. In contrast, FedRAP employs a bipartite personalized algorithm to manage data heterogeneity due to diverse user behaviors, considering both diversity and commonality of item information.

3 Federated Recommendation with Additive Personalization

Taking the fact that users' preferences for items are influenced by the users into consideration, in this research, we assume that the ratings on the clients are determined by the users' preferences, which are affected by both the user information and the item information. Thus, the user information is varied by different users, and the item information on each client should be similar, but not the same. Given partial rating data, the goal of FedRAP is to recommend the items users have not visited.

Notations. Let $\mathbf{R} = [r_1, r_2, \ldots, r_n]^T \in \{0,1\}^{n\times m}$ denote that the input rating data with n users and m items, where $r_i \in \{0,1\}^m$ indicates the i-th user's rating data. Since each client only have one user information as stated above, r_i also represents the i-th client's ratings. Moreover, we use $\mathbf{D}^{(i)} \in \mathbb{R}^{m\times k}$ to denote the local item embedding on the i-th client, and $\mathbf{C} \in \mathbb{R}^{m\times k}$ to denote the global item embedding. In the recommendation setting, one user may only rate partial items. Thus, we introduce $\mathbf{\Omega} = \{(i,j) : \text{the } i\text{-th user has rated the } j\text{-th item}\}$ to be the set of rated entries in $\mathbf{R}$.

3.1 Problem Formulation

To retain shared item information across users while enabling item personalization, we propose to use $\mathbf{C}$ to learn shared information, and employ $\mathbf{D}^{(i)}$ to capture item information specific to the i-th user on the corresponding i-th client, where $i = 1, 2, \ldots, n$. FedRAP then uses the summation $\mathbf{C}$ and $\mathbf{D}^{(i)}$ to achieve an additive personalization of items. Considering $\mathbf{R} \in \{0,1\}^{n\times m}$, we utilize the following formulation for the i-th client to map the predicted rating $\hat{r}_{ij}$ into $[0,1]$:

$$\min_{\mathbf{U},\mathbf{C},\mathbf{D}^{(i)}} \sum_{(i,j)\in\mathbf{\Omega}} -(r_{ij}\log\hat{r}_{ij} + (1-r_{ij})\log(1-\hat{r}_{ij})). \tag{1}$$

Here, $\hat{r}_{ij} = 1/(1+e^{-<\mathbf{u_i},(\mathbf{D^{(i)}}+\mathbf{C})j>})$ represents the predicted rating of the i-th user for the j-th item. Eq. (1) minimizes the reconstruction error between the actual rating r_i and predicted ratings $\hat{r}_{ij}$ based on rated entries indexed by $\mathbf{\Omega}$. To ensure the item information learned by $\mathbf{D}^{(i)}$ and $\mathbf{C}$ differs on the i-th client, we enforce the differentiation among them with the following equation:

$$\max_{\mathbf{C},\mathbf{D}^{(i)}} \sum_{i=1}^{n} ||\mathbf{D}^{(i)} - \mathbf{C}||_F^2. \tag{2}$$

Eqs. (1) and (2) describe the personalization model for each client, and aim to maximize the difference between $\mathbf{D}^{(i)}$ and $\mathbf{C}$. We note that in the early stages of learning, the item information learned by $\{\mathbf{D}^{(i)}\}_{i=1}^{n}$ and $\mathbf{C}$ may be inadequate for recommendation, and additive personalization may thus reduce performance. Thus, considering Eqs. (1), (2), and the aforementioned condition, we propose the following optimization problem for the client i in FedRAP:

$$\min_{\mathbf{U},\mathbf{C},\mathbf{D}^{(i)}} \sum_{i=1}^{n}(\sum_{(i,j)\in\mathbf{\Omega}} -(r_{ij}\log\hat{r}_{ij} + (1-r_{ij})\log(1-\hat{r}_{ij})) - \lambda_{(a,v_1)}||\mathbf{D}^{(i)} - \mathbf{C}||_F^2) + \mu_{(a,v_2)}||\mathbf{C}||_1, \tag{3}$$

where a is the number of training iterations. We employ the functions $\lambda_{(a,v_1)}$ and $\mu_{(a,v_2)}$ to control the regularization strengths, limiting them to values between 0 and v_1 or v_2, respectively. Here, v_1 and v_2 are constants that determine the maximum values of the hyperparameters. By incrementally boosting the regularization weights, FedRAP gradually transitions from full personalization to additive personalization concerning the item information. The L_1 norm on $\mathbf{C}$, denoted by $||\mathbf{C}||_1$, is employed to induce $\mathbf{C}$ sparsity, which eliminates unnecessary information from $\mathbf{C}$ and helps reduce the communication cost between the server and the client.

Algorithm 1 Federated Recommendation with Additive Personalization (FedRAP)

Input: $\mathbf{R}$, $\mathbf{\Omega}$, v_1, v_2, k, η, t_1, t_2
Initialize:$\mathbf{C}$
Global Procedure:
for each client index $i = 1, 2, \ldots, n$ **in parallel do**
  initialize $\mathbf{u}_i$, $\mathbf{D}^{(i)}$;
end for
for $a = 1, 2, \ldots, t_1$ **do**
  $S_a \leftarrow$ randomly select n_s from n clients
  for each client index $i \in S_a$ **in parallel do**
    download $\mathbf{C}$ from the server;
    $\mathbf{C}^{(i)}$, $\hat{r}_i \leftarrow$ ClientUpdate($\mathbf{u}_i$, $\mathbf{C}$, $\mathbf{D}^{(i)}$);
    upload $\mathbf{C}^{(i)}$ to the server;
  end for
  $\mathbf{C} \leftarrow \frac{1}{n_s}\sum_{i=1}^{n_s}\mathbf{C}^{(i)}$; ▷ Server Aggregation
end for
return: $\hat{\mathbf{R}} = [\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_n]^T$
ClientUpdate:
for $b = 1, 2, \ldots, t_2$ **do**
  calculate the partial gradients $\nabla\mathbf{u}_i$, $\nabla\mathbf{C}$ and $\nabla\mathbf{D}^{(i)}$ by differentiating Eq. (3);
  update $\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta\nabla\mathbf{u}_i$;
  update $\mathbf{C}^{(i)} \leftarrow \mathbf{C} - \eta\nabla\mathbf{C}^{(i)}$;
  update $\mathbf{D}^{(i)} \leftarrow \mathbf{D}^{(i)} - \eta\nabla\mathbf{D}^{(i)}$;
end for
$\hat{r}_i = \sigma(\langle\mathbf{u_i}, \mathbf{D^{(i)}} + \mathbf{C}\rangle)$;
return: $\mathbf{C}^{(i)}$, $\hat{r}_i$

3.2 Algorithm

Algorithm Design To address the objective described in Eq. (3), we employ an alternative optimization algorithm to train our model. As shown in Alg. 1, the overall workflow of the algorithm is summarized into several steps as follows. We start by initializing $\mathbf{C}$ at the server and $\mathbf{u}_i$ and $\mathbf{D}^{(i)}$ at their respective clients ($i = 1, 2, \ldots, n$). Each client is assumed to represent a single user. For every round, the server randomly selects a subset of clients, represented by S_a, to participate in the training. Clients initially receive the sparse global item embedding $\mathbf{C}$ from the server. They then invoke the function $\mathrm{ClientUpdate}$ to update their parameters accordingly with the learning rate η. Clients upload the updated $\{\mathbf{C}^{(i)}\}_{i=1}^{n_s}$ to the server for aggregation. The server then employs the aggregated $\mathbf{C}$ in the succeeding training round. Upon completion of the training process, FedRAP outputs the predicted ratings $\hat{\mathbf{R}}$ to guide recommendations. FedRAP upholds user privacy by keeping user-specific latent embeddings on clients and merely uploading the updated sparse global item

embedding to the server. This approach notably reduces both privacy risk and communication costs. Furthermore, FedRAP accommodates user behavioral heterogeneity by locally learning user-specific personalized models at clients. As such, even though user data varies due to individual preferences, FedRAP ensures that data within a user adheres to the i.i.d assumption.

Cost Analysis Regarding time cost, following Algorithm 1, the computational complexity on the i-th client for updating $\mathbf{u}_i$, $\mathbf{D}^{(i)}$, and $\mathbf{C}$ requires a cost of $\mathcal{O}(km)$. Once the server receives $\mathbf{C}$ from n clients, it takes $\mathcal{O}(kmn)$ for the server to perform aggregations. Consequently, the total computational complexity of Algorithm 1 at each iteration is $\mathcal{O}(kmn)$. In practice, the algorithm typically converges within 100 iterations. As for space cost, each client needs to store $\mathbf{u}_i \in \mathbb{R}^k$, $\mathbf{D}^{(i)} \in \mathbb{R}^{(m\times k)}$, and $\mathbf{C} \in \mathbb{R}^{(m\times k)}$, requiring $\mathcal{O}((2nm+n)k+nm)$ space. The server, on the other hand, needs to store the updated and aggregated global item embedding, requiring $\mathcal{O}((nm+m)k)$ space. Thus, FedRAP demands a total of $\mathcal{O}((3nm+n+m)k+nm)$ space.

3.3 Differential Privacy

Despite the fact that FedRAP only conveys the global item embedding $\mathbf{C}$ between the server and clients, there still lurks the potential risk of information leakage.

Theorem 1 (Revealing local gradient from consecutive $\mathbf{C}$) *If a server can obtain consecutive $\mathbf{C}^{(i)}$ returned by client i, it can extract the local gradient with respect to $\mathbf{C}$ from the latest update of the corresponding client.*

Proof 1 *If client i participates in two consecutive training rounds (denoted as a and $a+1$), the server can obtain the $\mathbf{C}^{(i)}$ returned by the client in both rounds, denoted as $\mathbf{C}^{(i)}_{(a)}$ and $\mathbf{C}^{(i)}_{(a+1)}$, respectively. According to Alg. 1, it can be deduced that $\mathbf{C}^{(i)}_{(a+1)} - \mathbf{C}^{(i)}_{(a)} = \eta\nabla\mathbf{C}^{(i)}_{(a+1)}$, where η is the learning rate. Since there must exist a constant equal to η, the server can consequently obtain the local gradient of the objective function with respect to $\mathbf{C}$ for client i during the $(a+1)$-th training round.*

Theorem 1 shows that a server, if in possession of consecutive $\mathbf{C}^{(i)}$ from a specific client, can derive the local gradient relative to $\mathbf{C}$ from the most recent client update, which becomes feasible to decipher sensitive user data from the client's local gradient [2]. This further illustrates that a client should not participate in training consecutively twice. Despite our insistence on sparse $\mathbf{C}$ and the inclusion of learning rate η in the gradient data collected by the server, the prospect of user data extraction from $\nabla\mathbf{C}^{(i)}_{(a+1)}$ persists. To fortify user privacy, we employ (ϵ, δ)-differential privacy protection [35] on $\mathbf{C}$, where ϵ measures the potential information leakage, acting as a privacy budget, while δ permits a small probability of deviating slightly from the ϵ-differential privacy assurance. To secure (ϵ, δ)-Differential Privacy, it's imperative for FedRAP to meet certain prerequisites. Initially, following each computation as outlined in Alg. 1, FedRAP secures a new gradient w.r.t. $\mathbf{C}$ on the i-th client, denoted by $\nabla\mathbf{C}^{(i)}$, which is limited by a pre-defined threshold τ:

$$\nabla\mathbf{C}^{(i)} \leftarrow \nabla\mathbf{C}^{(i)} \cdot \min\{1, \frac{\tau}{||\nabla\mathbf{C}^{(i)}||_F}\}. \tag{4}$$

Post gradient clipping in Eq. (4), $\nabla\mathbf{C}^{(i)}$ is utilized to update $\mathbf{C}$ on the client i. Upon receipt of all updated $\{\mathbf{C}^{(i)}\}_{i=1}^{n_s}$, the server aggregates to derive a new $\mathbf{C}$.

Theorem 2 (Sensitivity upper bound of $\mathbf{C}$) *The sensitivity of $\mathbf{C}$ is upper bounded by $\frac{2\eta\tau}{n_s}$.*

Proof 2 *Given two global item embeddings $\mathbf{C}$ and $\mathbf{C}'$, learned from two datasets that differ only in the data of a single user (denoted as u), respectively, we have:*

$$||\mathbf{C}-\mathbf{C}'||_F = ||\frac{\eta}{n_s}(\nabla\mathbf{C}^{(u)} - \nabla\mathbf{C}'^{(u)})||_F \leq \frac{\eta}{n_s}||\nabla\mathbf{C}^{(u)}||_F + \frac{\eta}{n_s}||\nabla\mathbf{C}'^{(u)}||_F \leq \frac{2\eta\tau}{n_s}. \tag{5}$$

Theorem 2 posits that the sensitivity of $\mathbf{C}$ cannot exceed $\frac{2\eta\tau}{n_s}$. To ensure privacy, we introduce noise derived from the Gaussian distribution $\mathcal{N}(0, e^2)$ to disrupt the gradient, where e is proportionate to the sensitivity of $\mathbf{C}$. As per [36], applying the Gaussian mechanism $(\frac{2\eta\tau}{n_s}, z)$ during a training

round ensures a privacy tuple of $(\frac{2\eta\tau}{n_s}, z)$. Following previous work [37], FedRAP leverages the composability and tail bound properties of moment accountants to determine ϵ for a given budget δ, thus achieving (ϵ, δ)-differential privacy.

4 Experiments

4.1 Datasets

A thorough experimental study has been conducted to assess the performance of the introduced FedRAP on five popular recommendation datasets: MovieLens-100K (ML-100K)[1], MovieLens-1M (ML-1M)[1], Amazon-Instant-Video (Video)[2], LastFM-2K[3] [38], and QB-article[4] [39]. The initial four datasets encompass explicit ratings that range between 1 and 5. As our task is to generate recommendation predictions on data with implicit feedback, any rating higher than 0 in these datasets is assigned 1. QB-article, which is an implicit feedback dataset, is derived from user interaction logs with articles. In each dataset, we include only those users who have rated at least 10 items.

4.2 Baselines

The efficacy of FedRAP is assessed against several cutting-edge methods in both centralized and federated settings for validation.

NCF [40]. It merges matrix factorization with multi-layer perceptrons (MLP) to represent user-item interactions, necessitating the accumulation of all data on the server. We set up three layers of MLP according to the original paper.

FedMF [2]. This approach employs matrix factorization in federated contexts to mitigate information leakage through the encryption of the gradients of user and item embeddings.

FedNCF [32]. This is the federated counterpart of NCF. It updates user embedding locally on each client and synchronizes the item embedding on the server for global updates.

PFedRec [34]. It aggregates item embeddings from clients on the server side and leverages them in the evaluation to facilitate bipartite personalized recommendations.

The implementations of NCF, FedMF and PFedRec are publicly available in corresponding papers. Although FedNCF does not provide the code, we have implemented a version based on the code of NCF and the paper of FedNCF. Moreover, we developed a centralized variant of FedRAP, termed **CentRAP**, to showcase the learning performance upper bound for FedRAP within the realm of personalized federated recommendation systems.

4.3 Experimental Setting

According to [40, 34], for each positive sample, we randomly selected 4 negative samples, and for each user, we chose 99 items with no interactions, ranking the test item among these 100 samples. We performed hyperparameter tuning for all methods, the v_1 parameter of our method FedRAP in the range $\{10^i | i = -6, \ldots, 0\}$, and the v_1 parameter of FedRAP in the range $\{10^i | i = -3, \ldots, 3\}$. Given that the second and third terms in Eq. 3 gradually come into effect during training, we pragmatically set the function $\lambda_{(a,v_1)} = \tanh(\frac{a}{10}) * v_1$ and $\mu_{(a,v_2)} = \tanh(\frac{a}{10}) * v_2$, where a is the number of iterations. For PFedRec and FedRAP, the maximum number of server-client communications and client-side training iterations were set to 100 and 10 respectively. For NCF, FedMF and FedNCF, we set the number of training iterations to 200. To be fair, all methods used fixed latent embedding dimensions of 32 and a batch size of 2048. We did not employ any pre-training for any of the methods, nor did we encrypt the gradients of FedMF. Following precedent [40, 41], we used a leave-one-out strategy for dataset split evaluation. All models were implemented using PyTorch[42], and experiments were conducted on a machine equipped with a 2.5GHz 14-Core Intel Core i9-12900H processor, a RTX 3070 Ti Laptop GPU, and 64GB of memory.

[1] `https://grouplens.org/datasets/movielens/`

[2] `http://jmcauley.ucsd.edu/data/amazon/`

[3] `https://grouplens.org/datasets/hetrec-2011/`

[4] `https://github.com/yuangh-x/2022-NIPS-Tenrec`

Evaluation Metrics. In the experiments, the prediction performances are evaluated by two widely used metrics in the recommendation field: *Hit Rate* (HR@K) and *Normalized Discounted Cumulative Gain* (NDCG@K). These criteria has been formally defined in [43]. In this work, we set $K = 10$, and repeat all experiments by five times, and report the average values and standard deviations.

4.4 Main Results and Comparisons

Fig. 1 illustrates the experimental results in percentages of all the comparing methods on the five real-world datasets, exhibiting that FedRAP outperforms the others in most scenarios and achieves best among all federated methods. The performance superiority probably comes from the ability of FedRAP on bipartite personalizations of both user information and item information. Moreover, the possible reason why FedRAP performs better than PFedRec is that FedRAP is able to personalize item information while retaining common information of items, thus avoiding information loss. CentRAP exhibits slightly better performance on all datasets compared to FedRAP, demonstrating the performance upper bound of FedRAP on the used datasets. In addition, to investigate the convergence of FedRAP, we compare the evaluations of all methods except CentRAP for each iteration during training on the ML-100K dataset. Because NCF, FedMF and FedNCF are trained for 200 iterations, we record evaluations for these methods every two iterations. Fig. 3 shows that on both HR@10 and NDCG@10, FedRAP achieves the best evaluations in less than 10 iterations and shows a convergence trend within 100 iterations. The experimental results demonstrate the superiority of FedRAP. However, since FedRAP is more complex than PFedRec, it needs more iterations to converge.

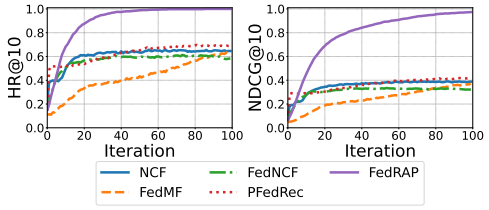


Figure 3: Convergence and efficiency comparison of five methods on the ML-100K dataset.

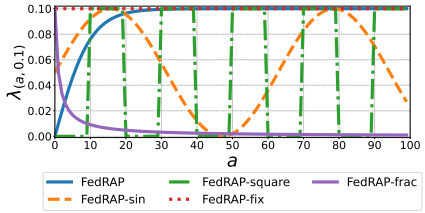


Figure 4: Different curriculum for $\lambda_(a, v_1)$ in Eq. (3) with $v_1 = 0.1$ and a to be the iteration.

4.5 Ablation Study

To examine the effects of FedRAP's components, we introduce several variants: FedRAP-C, FedRAP-D, FedRAP-No, FedRAP-L2, FedRAP-fixed, FedRAP-sin, FedRAP-square, and FedRAP-frac, defined as follows:

- **FedRAP-C**: Uses a common item embedding $\mathbf{C}$ for all users, ignoring personalization.
- **FedRAP-D**: Creates user-specific item embeddings $\mathbf{D}^{(i)}$, without shared item information.
- **FedRAP-No**: Removes item diversity constraint on $\mathbf{C}$, i.e., $||\mathbf{C}||_1$ is omitted.
- **FedRAP-L2**: Uses Frobenius Norm, $||\mathbf{C}||_F^2$, instead of $||\mathbf{C}||_1$ for constraint.
- **FedRAP-fixed**: Fixes $\lambda_{(a,v_1)}$ and $\mu_{(a,v_2)}$ to be two constants v_1 and v_2, respectively.
- **FedRAP-sin**: Applies $\lambda_{(a,v_1)} = \sin(\frac{a}{10}) * v_1$ and $\mu_{(a,v_2)} = \sin(\frac{a}{10}) * v_2$.
- **FedRAP-square**: Alternates the value of $\lambda_{(a,v_1)}$ and $\mu_{(a,v_2)}$ between 0 and v_1 or v_2 every 10 iterations, respectively.
- **FedRAP-frac**: Utilizes $\lambda_{(a,v_1)} = \frac{v_1}{a+1}$ and $\mu_{(a,v_2)} = \frac{v_2}{a+1}$.

Here, a denotes the number of iterations and v is a constant. FedRAP-C and FedRAP-D validate the basic assumption of user preference influencing ratings. FedRAP-No and FedRAP-L2 assess the impact of sparsity in $\mathbf{C}$ on recommendations. The last four variants explore different weight curricula on FedRAP's performance. We set v_1 and v_2 to 0.1 for all variants, and Fig. 4 depicts the hyperparameter trends $\lambda_{(a,v_1)}$ of for the final four variants. The trend of $\mu_{(a,v_2)}$ is the same as the corresponding $\lambda_{(a,v_1)}$ of each variant.

Ablation Study of FedRAP To verify the effectiveness of FedRAP, we train all variants as well as FedRAP on the ML-100K dataset. In addition, since PFedRec is similar to FedRAP-D, we also

compare it together in this case. As shown in Fig. 5, we plot the trend of HR@10 and NDCG@10 for these six methods over 100 iterations. From Fig. 5, we can see that FedRAP-C performs the worst, followed by FedRAP-D and PFedRec. These indicate the importance of not only personalizing the item information that is user-related, but also keeping the parts of the item information that are non-user-related. In addition, both FedRAP-C and FedRAP-D reach convergence faster than FedRAP. Because FedRAP-C trains a global item embedding $\mathbf{C}$ for all clients, and FedRAP-D does not consider $\mathbf{C}$, while FedRAP is more complex compared to these two algorithms, which explains the slow convergence speed of FedRAP because the model is more complex. The figure also shows that the performance of FedRAP-D and PFedRec is similar, but the curve of FedRAP-D is smoother, probably because both have the same assumptions and similar theories, but we add regularization terms to FedRAP. The results of FedRAP-No, FedRAP-L2 and FedRAP are very close, where the performance of FedRAP is slightly better than the other two as seen in Fig. 5(b). Thus, we choose to use L_1 norm to induce $\mathbf{C}$ to become sparse. Also, a sparse $\mathbf{C}$ helps reduce the communication overhead between the server and clients.

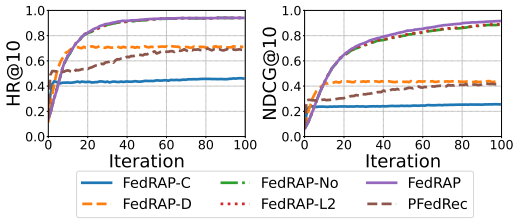


Figure 5: **Ablation study** investigating the effectiveness of FedRAP on the ML-100K dataset.

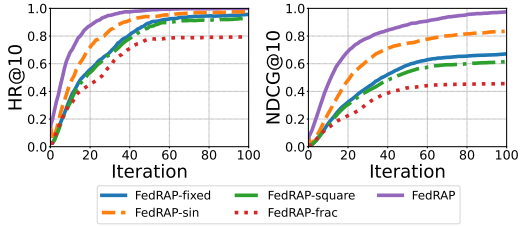
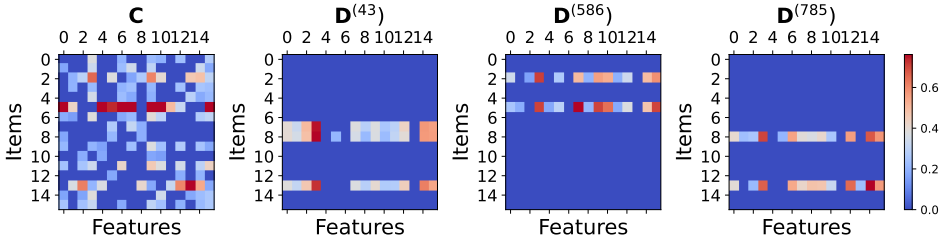
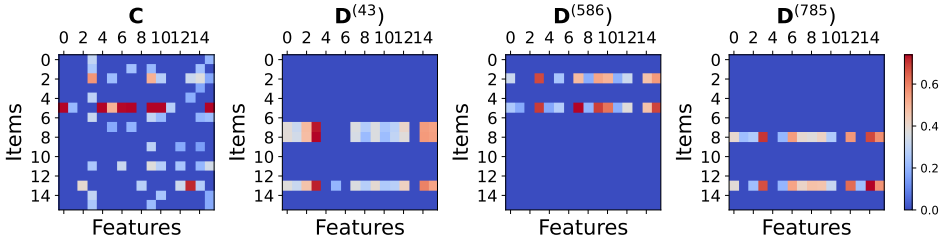


Figure 6: **Comparison of different curricula** for $\lambda_{(a,v_1)}$ and $\mu_{(a,v_2)}$ in FedRAP on the ML-100K dataset.

Ablation Study on Curriculum In this section, we investigate the suitability of additive personalization in the initial training phase by analyzing the performance of FedRAP and its four weight variations (FedRAP-fixed, FedRAP-sin, FedRAP-square, and FedRAP-frac) on HR@10 and NDCG@10 in the ML-100K dataset. Fig. 6 shows that FedRAP-fixed, with constant weights, underperforms FedRAP, which adjusts weights from small to large. This confirms that early-stage additive personalization leads to performance degradation, as item embeddings have not yet captured enough useful information. Furthermore, FedRAP-frac, which reduces weights from large to small, performs the worst, while FedRAP performs the worst, reiterating that early-stage additive personalization can decrease performance.



(a) FedRAP-L2



(b) FedRAP

Figure 7: Sparse global matrix $\mathbf{C}$ and row-dense personalized matrix $\mathbf{D}^{(i)}$ for item embedding.

Study on Sparsity of $\mathbf{C}$ Since FedRAP only exchanges $\mathbf{C}$ between server and clients, retaining user-related information on the client side, we further explore the impact of sparsity constraints on $\mathbf{C}$ in FedRAP by comparing the data distribution of $\mathbf{C}$ and $\{\mathbf{D}^{(i)}\}_{i=1}^{n}$ learned in the final iteration when training both FedRAP and FedRAP-L2 on the ML-100K dataset. We select three local item embeddings for comparison: $\mathbf{D}^{(43)}$, $\mathbf{D}^{(586)}$, and $\mathbf{D}^{(786)}$, along with the global item embedding $\mathbf{C}$. For visualization, we set latent embedding dimensions to 16 and choose 16 items. Fig. 7 reveals that FedRAP's global item embedding $\mathbf{C}$ is sparser than that of FedRAP-L2. We also notice that 4.19% of FedRAP's $\mathbf{C}$ parameters exceed 1e-1 and 67.36% exceed 1e-2, while for FedRAP-L2, these values are 8.07% and 88.94%, respectively. This suggests that FedRAP's sparsity constraint reduces communication overhead. Additionally, both methods learn similar $\mathbf{D}^{(i)}$ (i = 46, 586, 785), with notable differences between distinct $\mathbf{D}^{(i)}$. This supports our hypothesis that user preferences influence item ratings, leading to data heterogeneity in federated settings and validating FedRAP's ability to learn personalized information for each user.

To examine factors affecting FedRAP's performance, we designed ML-1M dataset variations with different sparsity levels by selecting specific numbers of users or items. Table 1 displays FedRAP's performance on these datasets, and shows that as sparsity increases, performance declines more significantly. Additionally, with similar sparsity, increasing user count leads to slightly reduced performance. Datasets with a fixed number of items consistently underperform compared to those with equal user counts. This table indicates that FedRAP's effectiveness is influenced by dataset sparsity and the counts of users and items.

Table 1: Sparsity of various subsets in ML-1M and FedRAP's fine-grained performance (HR@10 and NDCG@10) on them.

Subset	Sparsity	HR@10	NDCG@10
w/ 500 users	81.33%	1.0000	0.9446
	95.49%	1.0000	0.8227
	98.62%	0.9020	0.5686
w/ 1000 users	85.87%	1.0000	0.9064
	97.52%	0.9990	0.7118
	98.97%	0.8260	0.5207
w/ 1000 items	87.62%	0.9851	0.8931
	94.86%	0.9398	0.7759
	96.85%	0.9186	0.7003

Table 2: FedRAP vs. FedRAP-noise (FedRAP with injected noise for **differential privacy**)

Dataset	Metrics	FedRAP	FedRAP-noise	Degradation
ML-100K	HR@10	0.9709	0.9364	↓ 3.55%
	NDCG@10	0.8781	0.8015	↓ 8.72%
ML-1M	HR@10	0.9324	0.9180	↓ 1.54%
	NDCG@10	0.7187	0.7035	↓ 2.11%
Video	HR@10	0.4653	0.4191	↓ 9.93%
	NDCG@10	0.2817	0.2645	↓ 6.11%
LastFM-2K	HR@10	0.2329	0.2268	↓ 2.62%
	NDCG@10	0.1099	0.1089	↓ 0.91%
QB-article	HR@10	0.5398	0.5020	↓ 7.00%
	NDCG@10	0.2498	0.2475	↓ 4.73%

FedRAP with Differential Privacy To protect user privacy, we introduce noise from the Gaussian distribution to the gradient w.r.t. $\mathbf{C}$ on clients and compare the performance of FedRAP with and without noise. In this work, we set $\tau = 0.1$, and fix the noise variance $z = 1$ for Gaussian Mechanism following previous work [36]. We use the Opacus libaray [42] to implement the differential privacy. Table 2 presents results for these approaches on five datasets, showing reduced FedRAP performance after noise addition. However, as Fig 1 indicates, it still outperforms benchmark methods. Moreover, since FedRAP only requires the transmission of $\mathbf{C}$ between the server and clients, adding noise solely to $\mathbf{C}$ is sufficient for ensuring privacy protection. This results in a mitigated performance decline due to differential privacy compared to adding noise to all item embeddings.

5 Conclusions

In this paper, based on the assumption that user's ratings of items are influenced by user preferences, we propose a method named FedRAP to make bipartite personalized federated recommendations, i.e., the personalization of user information and the additive personalization of item information. We achieves a curriculum from full personalization to additive personalization by gradually increasing the regularization weights to mitigate the performance degradation caused by using the additive personalization at the early stages. In addition, a sparsity constraint on the global item embedding to remove useless information for recommendation, which also helps reduce the communication cost. Since the client only uploads the updated global item embedding to the server in each iteration, thus FedRAP avoids the leakage of user information. We demonstrate the effectiveness of our model through comparative experiments on four widely-used real-world recommendation datasets, and numerous ablation studies. Also, due to the simplicity of FedRAP, it would be interesting to explore its applications in other federated scenarios.

References

[1] Debashis Das, Laxman Sahoo, and Sujoy Datta. A survey on recommendation system. *International Journal of Computer Applications*, 160(7), 2017.

[2] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. Secure federated matrix factorization. *IEEE Intelligent Systems*, 36(5):11–20, 2020.

[3] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[4] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[5] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[6] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[7] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[8] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[10] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.

[11] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[12] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems*, 36(5):21–30, 2020.

[13] Feng Liang, Weike Pan, and Zhong Ming. Fedrec++: Lossless federated recommendation with explicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4224–4231, 2021.

[14] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, J Keith Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[15] Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pages 17716–17758. PMLR, 2022.

[16] Zareen Alamgir, Farwa K Khan, and Saira Karim. Federated recommenders: methods, challenges and future. *Cluster Computing*, 25(6):4075–4096, 2022.

[17] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[18] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[19] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized federated learning: An attentive collaboration approach. 2020.

[20] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[21] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[22] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting. *IEEE Transactions on Cognitive Communications and Networking*, 7(4):1078–1088, 2021.

[23] Weituo Hao, Mostafa El-Khamy, Jungwon Lee, Jianyi Zhang, Kevin J Liang, Changyou Chen, and Lawrence Carin Duke. Towards fair federated learning with zero-shot data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3310–3319, 2021.

[24] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[25] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[26] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[27] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pages 2089–2099. PMLR, 2021.

[28] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[29] Adrian Flanagan, Were Oyomno, Alexander Grigorievskiy, Kuan E Tan, Suleiman A Khan, and Muhammad Ammad-Ud-Din. Federated multi-view matrix factorization for personalized recommendations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 324–347. Springer, 2020.

[30] Sichun Luo, Yuanzhang Xiao, and Linqi Song. Personalized federated recommendation via joint representation learning, user clustering, and model adaptation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4289–4293, 2022.

[31] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Yali Jiang, Fangzhao Wu, and Lizhen Cui. A survey on federated recommendation systems. *arXiv preprint arXiv:2301.00767*, 2022.

[32] Vasileios Perifanis and Pavlos S Efraimidis. Federated neural collaborative filtering. *Knowledge-Based Systems*, 242:108441, 2022.

[33] Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

[34] Chunxu Zhang, Guodong Long, Tianyi Zhou, Peng Yan, Zijian Zhang, Chengqi Zhang, and Bo Yang. Dual personalization on federated recommendation. *arXiv preprint arXiv:2301.08143*, 2023.

[35] Lorenzo Minto, Moritz Haller, Benjamin Livshits, and Hamed Haddadi. Stronger privacy for federated collaborative filtering with implicit feedback. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 342–350, 2021.

[36] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[37] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[38] Iv'an Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

[39] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun YU, Bo Hu, Zang Li, Yu Xu, and Xiaohu Qie. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[40] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[41] Anonymous. Dual personalization for federated recommendation on devices. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. under review.

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[43] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1661–1670, 2015.