8-2020

Learning transferrable parameters for long-tailed sequential user behavior modeling

Jianwen YIN
Zhejiang University

Chenghao LIU
Singapore Management University, chliu@smu.edu.sg

Weiqing WANG
Monash University

Jianling SUN
Zhejiang University

Steven C. H. HOI
Singapore Management University, chhoi@smu.edu.sg

Citation

YIN, Jianwen; LIU, Chenghao; WANG, Weiqing; SUN, Jianling; and HOI, Steven C. H.. Learning transferrable parameters for long-tailed sequential user behavior modeling. (2020). *KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, San Diego, CA, August 22-27*. 359-367. Research Collection School Of Computing and Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/5890

Learning Transferrable Parameters for Long-tailed Sequential User Behavior Modeling

Jianwen Yin [1,4,*], Chenghao Liu[2,*], Weiqing Wang[3], Jianling Sun[1,4], Steven C.H. Hoi[2,5]
[1] Zhejiang University, [2]Singapore Management University,[3] Monash University
[4]Alibaba-Zhejiang University Joint Institute of Frontier Technologies, [5] Salesforce Research Asia
yinjw@zju.edu.cn,chliu@smu.edu.sg,Teresa.Wang@monash.edu,sunjl@zju.edu.cn,shoi@salesforce.com

ABSTRACT

Sequential user behavior modeling plays a crucial role in online user-oriented services, such as product purchasing, news feed consumption, and online advertising. The performance of sequential modeling heavily depends on the scale and quality of historical behaviors. However, the number of user behaviors inherently follows a long-tailed distribution, which has been seldom explored. In this work, we argue that focusing on tail users could bring more benefits and address the long tails issue by learning transferrable parameters from both optimization and feature perspectives. Specifically, we propose a gradient alignment optimizer and adopt an adversarial training scheme to facilitate knowledge transfer from the head to the tail. Such methods can also deal with the cold-start problem of new users. Moreover, it could be directly adaptive to various well-established sequential models. Extensive experiments on four real-world datasets verify the superiority of our framework compared with the state-of-the-art baselines.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Sequential User Behavior Modeling; Long-tailed Distribution; Gradient Alignment; Adversarial Training

ACM Reference Format:
Jianwen Yin [1,4,*], Chenghao Liu[2,*], Weiqing Wang[3], Jianling Sun[1,4], Steven C.H. Hoi[2,5]. 2020. Learning Transferrable Parameters for Long-tailed Sequential User Behavior Modeling. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403078

1 INTRODUCTION

With the rapid development of the Internet, the applications of user sequential scenarios have become essential and pervasive, such as e-commerce system, news/articles suggestion, and click-through rate (CTR) prediction [12, 13, 30–32]. In such applications, each user's behaviors can be represented as sequences in chronological orders, and his/her future behaviors can be predicted with given historical behaviors. Modeling users' complex sequential behaviors is challenging and critically important for providing a personalized recommendation in real-world applications [29].

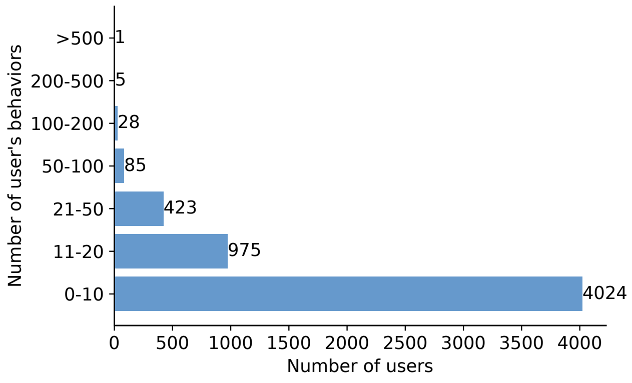


Figure 1: Histogram of the number of users over the number of user's behaviors of the Amazon Music dataset.

Generally, the performance of sequential user behavior modeling heavily depends on the scale and quality of historical behaviors. To achieve the desired performance, it is required to have enough historical behaviors for each user for sufficient modeling. In real-world applications, however, the number of user's behaviors inherently follows a long-tailed distribution as shown in 1, in which the number of historical behaviors of per user varies significantly from hundreds or thousands for head users to as few as one for tail users [1, 24]. Although existing approaches for sequential user behavior modeling achieve promising results on those few data-rich head users, they leave many data-poor tail users ill-served. In fact, this common property of real-world datasets encourages a skewed user behavior prediction policy where many tail users are modeled far worse than others.

In contrast, focusing on tail users can bring more benefits: (1) Although head users take a large share of total user-item interactions, the number of tail users is much larger than that of head users. Improving the performance of tail users can significantly increase the retention rate and scale of users, thereby producing massive profits. (2) Compared with the performance over head users where is good enough and has limited room for improvement, tail users embrace relatively large improvement, which means the endeavor to explore the tail users can bring much more improvements. (3) Attention on the tail users can also boost the performance on the

[*] denotes equal contribution.
Chenghao Liu and Jianling Sun are corresponding authors. Steven C.H. Hoi is currently with Salesforce Research Asia and on leave from Singapore Management University.

head. The infrequent patterns lie in the tail could provide a complementary understanding of the head. (4) In practice, when a learned model is deployed in an online setting, it requires to deal with new users who have no interactions available during training phase [7]. Focusing on those more diverse and noisy tail users is helpful for learning a robust model that can generalize well on new users.

Given these issues with conventional sequential models, how can we learn a model that is focusing on improving recommendation accuracy for badly-modeled tail users? While it focuses on the tail, could we enhance the general performance of both head users and tail users? Could it improve the performance of new users simultaneously?

In this paper, we aim to address the long-tailed sequential user behavior modeling by learning transferrable parameters. The key idea is that some internal information is more transferrable across users, while others can cause interference. Encouraging the expression of such information during training can improve the performance of tail users, thereby generating an unbiased model which can perform well on both long and tail users. Such a model can also deal with the cold-start problem of new users that are common in real-world applications. Specifically, we address the problem of learning transferrable parameters from both the optimization perspective and feature perspective. In **optimization perspective**, we develop a gradient alignment optimization method, which could maximize transfer while minimizing interference across users. In the learning process, we update a mini-batch of users with gradient descent. For each pair of users, the gradient angle implies the transferrable ability between these two users. The more consistent the directions of the gradients, the more knowledge they can share. Based on this, we impose an auxiliary loss to maximize the dot product between the gradients generated by different users. In **feature perspective**, we introduce a discriminator which takes a sequential embedding as input and classifies whether it belongs to a head user or a tail user. As the model achieves equilibrium, the discriminator cannot well differentiate head users from tail users. Consequently, head users are mixed with tail users in the embedding space. In this way, the bias caused by a large amount of data from head users could be reduced, thus facilitating knowledge transfer from the head to the tail.

The main contributions of this work are summarized as follows:

- To the best of our knowledge, this is the first work that addresses the long-tailed sequential user behavior modeling. We argue that focusing on tail users can bring more benefits and achieve this by learning transferrable parameters.
- To learn transferrable parameters, we propose a gradient alignment optimizer to transfer knowledge across users from the optimization perspective. Moreover, we introduce an adversarial training method to learn frequency-agnostic sequential embedding, which facilitates knowledge transfer from the feature perspective. The proposed method could be adaptive to various well-established sequential models, such as GRU4REC [8], CASER [26] and SASR[9].
- We conduct extensive experiments by evaluating the proposed method on real-world datasets, and show that it outperforms the existing state-of-the-art baselines for sequential user behavior modeling task.

2 RELATED WORK

In this section, we will review several lines of works closely related to this paper, including sequential user behavior modeling, gradient alignment, and adversarial training.

2.1 Sequential User Behavior Modeling

User behavior modeling, which captures users' preferences from behavior data, is critically important since it contributes significant improvement for real-world applications. Researchers have proposed various approaches, from traditional collaborative filtering models [10, 15] to deep representation learning models [18, 37]. These models focus on mining the static relationships between users and items, ignoring the dynamics of users' preferences implied in sequential interactions.

Nowadays, sequential user behavior modeling has attracted considerable attention due to its superiority in capturing item-to-item sequential patterns. Early work [19, 28, 38] mostly focus on Markov chain [17] models. [23] employs Markov decision processes in the recommender system to provide recommendations using sequential information. With the success of deep learning, researchers adopt (deep) neural network [8, 11, 14, 34] to model the sequential dynamics. Particularly, [8] uses Gated Recurrent Units to encode previous behaviors into a hidden vector for the recommendation. Besides that, [26] proposes a sequential model to learn sequential patterns using both horizontal and vertical convolutional filters. Recently, self-attention [27] attains promising results in various NLP tasks. [9] firstly adopts the self-attention mechanism for sequential user behavior modeling, achieving state-of-the-art performance on the sequential recommendation.

Although the aforementioned methods achieve satisfactory results on sequential user behavior modeling task, they ignore the problem of long-tailed distribution, which may cause performance degradation for tail users. In this paper, we address this issue by learning transferrable parameters.

2.2 Gradient Alignment

The idea of gradient alignment has been well studied in various fields. Leap [2] utilizes gradient alignment to transfer knowledge across the learning process. [20] attempts to solve the continual learning problem by considering a temporally symmetric tradeoff between transfer and interference, which is implemented by encouraging gradient alignment across examples. More recently, [35] presents the Lookahead optimization method, which improves learning stability and achieves faster convergence. In each step, the updating rule encourages the model parameters towards the aligned direction of gradients generated by different mini-batches. Different from previous works, we use gradient alignment to learn transferrable parameters in sequential user behavior modeling task.

2.3 Adversarial Training

Adversarial training is a well-studied problem [5, 25], in which two or more models learn together by pursuing competing goals. A representative work of adversarial training is Generative Adversarial Nets [4, 21], in which a discriminator and a generator compete with each other: the generator aims to generate samples similar to the real ones from random noise, and the discriminator aims to distinguish between the generated and the real samples. The networks are trained jointly using backpropagation on the prediction loss in a mini-max fashion: update generator to minimize the loss while also updating discriminator to maximize the loss. In this work, we try to learn frequency-agnostic sequential embedding with the help of adversarial training. Specifically, we introduce a discriminator to differentiate sequential embeddings of head users and tail users while the prediction model aims to fool the discriminator to misclassify users and minimize the prediction loss simultaneously. In this way, the bias caused by the long-tail distribution of the number of user behaviors could be eliminated. A similar optimization approach has shown promising results in the neural word embedding literature [3].

3 PROPOSED METHODOLOGY

In this section, we first give the notations and preliminaries of sequential user behavior modeling. Then we present how to learn transferrable parameters across users from optimization perspective and feature perspective, respectively. The structure of the proposed method is illustrated in Figure 2.

3.1 Problem Formulation

Assume we have a set of users $\mathbf{U} = \{u_1, u_2, \cdots, u_{|\mathbf{U}|}\}$ and a universe of items $\mathbf{I} = \{i_1, i_2, \cdots, i_{|\mathbf{I}|}\}$. Each user u is associated with a sequence of items sorted by time, which is represented as $\mathrm{S}^u = \{S_1^u, S_2^u, \cdots, S_t^u, \cdots, S_{|\mathrm{S}^u|}^u\}$, where $S_t^u \in \mathbf{I}$ denotes a user u ever interacted with the item S_t^u at time t. The objective is to seek a prediction model such that for a given prefix item subsequence $\mathrm{S}^{(u,t)} = \{S_{t-L}^u, S_{t-L+1}^u, \cdots, S_{t-1}^u\}$ including the most recent L items before time t, it can generate a ranking score for all items:

$$\mathrm{y}_u = f(\mathrm{S}^{(u,t)}; \theta). \tag{1}$$

The prediction model f is a composition of the sequence embedding function ϕ and a score function h, which can be written as $f(\cdot) = h(\phi(\cdot))$. We denote the parameters of prediction model f as θ. The binary cross entropy loss is often used as the optimization target [9, 26]:

$$\mathcal{L} = \sum_u \sum_{j \in \mathrm{S}^u} [-\log(\sigma(\mathrm{y}_{u,j})) - \sum_{k \notin \mathrm{S}^u} \log(1 - \sigma(\mathrm{y}_{u,k}))], \tag{2}$$

where σ is the sigmoid function. Since the number of non-interactive items is relatively large, we follow the negative sampling strategy [19, 26] to choose the negative instances. The major notations in this paper are listed in table 1.

Table 1: List of notations.

Notation	Meaning
$\mathbf{U}, \mathbf{I}$	the sets of users and items
S^u	the item sequence of user u
f, θ	the prediction model and parameters
f_d, θ_d	the discriminator and parameters
y_u	the prediction scores of user u
α, β	the learning rates of inner and outer update
k	the inner update times
λ	the adversarial hyper-parameter

The problem of long tails makes existing sequential models perform poorly on tail users. To address this problem, we formulate each user as an individual task and then learn transferrable parameters across different tasks. For each user u, we extract every L successive items as input and their next one item as the target from S^u. Then we can get a corresponding training task $\mathcal{T}_u = \{(\mathrm{S}^{(u,t)}, S_t^u) | t \in \{L+1, L+2, ..., |\mathrm{S}^u|\}\}$. Given a set of training tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{|\mathbf{U}|}\}$, we aim to strike a balance between the performance of data-rich head users and data-poor tail users by exploiting transferrable model parameters.

3.2 Gradient Alignment

The intuition behind our method is that when training with multiple tasks, some internal information are more transferrable across tasks while others can cause interference [20]. Taking two users i and j as examples, the corresponding training tasks are $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. At each iteration, we sample K subsequences from $\mathcal{T}_i$ and $\mathcal{T}_j$ separately for update. These two mini-batches are denoted as $\mathcal{D}_i$ and $\mathcal{D}_j$. Then, we can define operational measures of transfer and interference between these two distinct mini-batches. Formally, the concept of transfer is formulated as:

$$\frac{\partial \mathcal{L}(\mathcal{D}_i; \theta)}{\partial \theta} \cdot \frac{\partial \mathcal{L}(\mathcal{D}_j; \theta)}{\partial \theta} > 0, \tag{3}$$

where $\cdot$ is the dot product operator. It indicates that solving the task of user i will facilitate the learning process of the task of user j, and vice versa (Figure 3(a)). In contrast, the concept of interference is formulated as:

$$\frac{\partial \mathcal{L}(\mathcal{D}_i; \theta)}{\partial \theta} \cdot \frac{\partial \mathcal{L}(\mathcal{D}_j; \theta)}{\partial \theta} < 0. \tag{4}$$

It implies that learning the task of user i can impede the learning process of user j and vice versa (Figure 3(b)). The potential for transfer is maximized when weight sharing is maximized while potential for interference is maximized when weight sharing is minimized. Since the tail users have limited data for training, encouraging the emergence of such transferrable parameters naturally transfer the knowledge from data-rich head users to data-poor tail users. Moreover, the transferrable parameters enjoy a better generalization capability, which facilitate the prediction of new users.

To maximize transfer and minimize interference, we incur an auxiliary loss to the objective, which can bias the learning process to that direction. According to Eq. (3) and (4), we evaluate the gradients of randomly sampled mini-batches from different users. By maximizing the inner product between the gradients of different mini-batches, we promote the model transferability by sharing parameters where gradient directions align. To this end, we can express our optimization objective as:

$$\begin{aligned}\theta =& arg \min_\theta \mathbb{E}_{\mathcal{D}_i \& \mathcal{D}_j} \\ & \left[\mathcal{L}(\mathcal{D}_i; \theta) + \mathcal{L}(\mathcal{D}_j; \theta) - \alpha \frac{\partial \mathcal{L}(\mathcal{D}_i; \theta)}{\partial \theta} \cdot \frac{\partial \mathcal{L}(\mathcal{D}_j; \theta)}{\partial \theta} \right].\end{aligned} \tag{5}$$

Overall, the first two terms focus on the minimum of the expected loss over tasks, while the last term enables knowledge transfer across users by maximizing the inner product between different gradients.

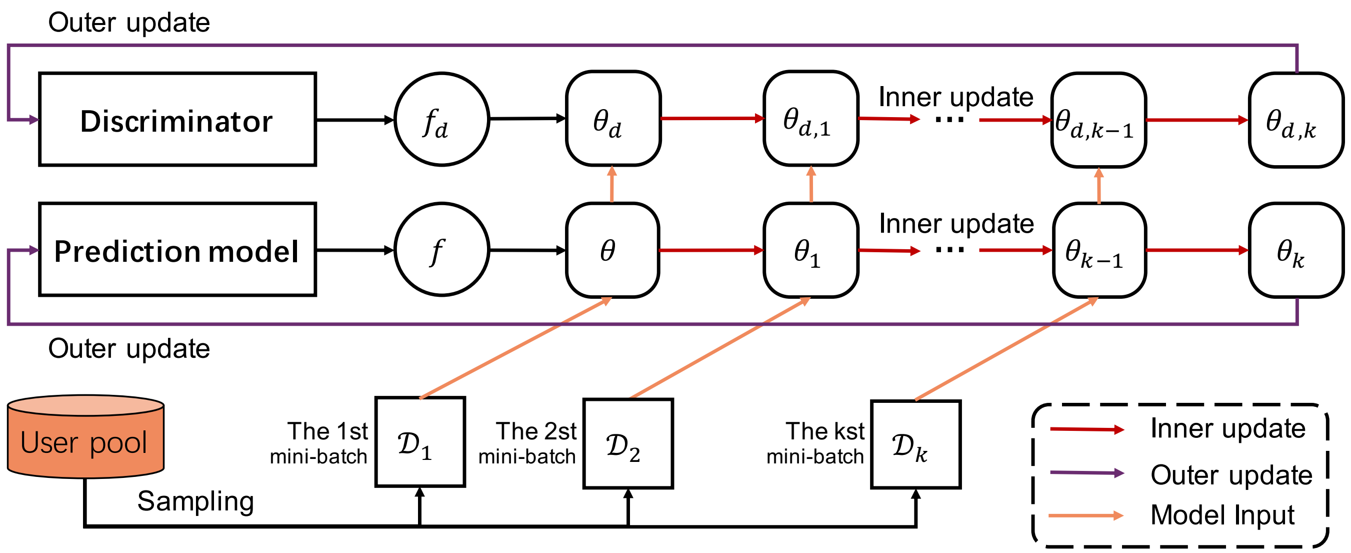


Figure 2: The structure of the proposed method. Each iteration consists of k inner updates and one outer update. In each inner update phase (the red arrows), we select a user and sample a mini-batch of L-length subsequences from corresponding task for training. The parameters of the prediction model and the discriminator are updated with gradient descent. In outer update phase (the purple arrows), we update the prediction model and the discriminator in the direction of difference between the initial parameters and updated parameters.

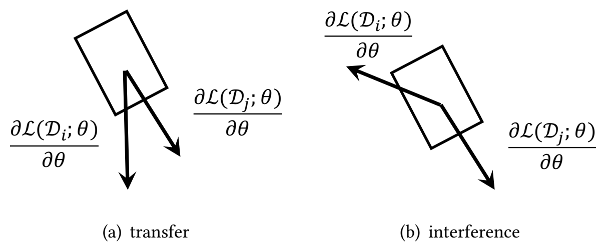

$\frac{\partial \mathcal{L}(\mathcal{D}_i;\theta)}{\partial \theta}$
$\frac{\partial \mathcal{L}(\mathcal{D}_j;\theta)}{\partial \theta}$
(a) transfer
$\frac{\partial \mathcal{L}(\mathcal{D}_i;\theta)}{\partial \theta}$
$\frac{\partial \mathcal{L}(\mathcal{D}_j;\theta)}{\partial \theta}$
(b) interference

Figure 3: (a) A depiction of transfer across users. (b) A depiction of interference across users.

However, directly optimizing the above objective requires to explicitly compute second derivatives, which is quite expensive. Motivated by [16], we approximate the objective in (5) with a first order Taylor expansion to reduce computational overhead. Specifically, the training process is divided into two phases: inner update and outer update. In inner update phase, we perform k gradient updates with learning rate α. At each time, we randomly select one user and sample a mini-batch from corresponding task for update:

$$\theta_k = \theta_{k-1} + \alpha \frac{\partial \mathcal{L}(\mathcal{D}_k; \theta_{k-1})}{\partial \theta_{k-1}}. \tag{6}$$

These inner updates start with parameters θ and result in parameters θ_k. Then we enter outer update phase: updating the model parameters θ in the direction $\theta - \theta_k$:

$$\theta = \theta - \beta * (\theta - \theta_k), \tag{7}$$

where β is the learning rate. According to taylor's theorem, the above update process is approximately equivalent to optimize the following objective:

$$\theta = arg \min_{\theta} \mathbb{E}_{\mathcal{D}_1,\cdots,\mathcal{D}_k} \left[\sum_{i=1}^{k} \left[\mathcal{L}(\mathcal{D}_i;\theta) - \frac{1}{2} \sum_{j=1}^{i-1} \alpha \frac{\partial \mathcal{L}(\mathcal{D}_i;\theta)}{\partial \theta} \cdot \frac{\partial \mathcal{L}(\mathcal{D}_j;\theta)}{\partial \theta} \right] \right], \tag{8}$$

where $\mathcal{D}_1, \cdots, \mathcal{D}_s$ are mini-batches sampled from different tasks. Compared to the original objective in (5), Eq. (8) also contains the transferability-promoting terms. The difference lies in the importance of different users. Eq. (8) assigns high importance to the early chosen user in the inner update. As we randomly choose user at each inner update, Eq. (8) does not bring any bias towards users.

3.3 Adversarial Training

Although gradient alignment could encourage knowledge transfer across users, the prediction model, especially for the sequence embedding function, will be largely dominated by the head users which have more data available for training. More severely, the sequential embeddings of head users and tail users tend to lie in different spaces. It will consequently limit the performance of prediction using the embeddings. To address this issue, we develop an adversarial training method [4] to mix head users and tail users into a common embedding space and thus produce frequency-agnostic sequential embedding. Specifically, we adopt a discriminator to categorize users into two classes: head or tail. We hope that the discriminator optimizes its parameters to maximize its classification accuracy, while the prediction model is optimized towards a low training loss as well as fooling the discriminator to misclassify head users and tail users. When the whole optimization process converges, the discriminator cannot well classify head users and tail users. In this way, the bias caused by the large amount of data from head users could be reduced, thus facilitating knowledge transfer from the head to the tail.

To begin with, we divide users into two parts based on the number of historical behaviors and use $R \in \{0, 1\}$ to indicate this property: $R = 1$ for head users; $R = 0$ for tail users. Let f_d and θ_d denote the discriminator and its parameters respectively, then the loss of the discriminator $\mathcal{L}_d$ can be defined as:

$$\begin{aligned}\mathcal{L}_d(\mathcal{D};\theta,\theta_d) =& R\log f_d(\phi(\mathcal{D};\theta);\theta_d) \\ &+ (1-R)\log(1-f_d(\phi(\mathcal{D};\theta);\theta_d).\end{aligned} \tag{9}$$

Following the principle of adversarial training, we develop a minimax objective to train the prediction model and the discriminator as below:

$$\min_{\theta,\theta_d}\max_{\theta_d}\mathcal{L}(\mathcal{D};\theta) - \lambda\mathcal{L}_d(\mathcal{D};\theta,\theta_d), \tag{10}$$

where λ is a hyper-parameter to trade off the two loss terms.

Now we summarize the training process of our method in Algorithm 1. In each iteration, we perform k inner updates and one outer update. the proposed method requires $k \geq 2$, where the update depends on the higher-order derivatives of the loss function. Otherwise, if we set k to be 1, it will degrade to a classic joint optimization over all tasks. It is worth mentioning that our method is model agnostic and can be applied to any existing models for sequential recommendation, such as GRU4REC [8], CASER [26] and SASR [9], which are used for our empirical study.

Algorithm 1 Proposed Algorithm

```
Input: U: the set of existing users
Input: $\mathcal{T}$: the set of training tasks
Input: $\alpha, \beta$: the learning rates of inner and outer update
Input: $k$: the inner update times
Input: $\lambda$: the adversarial hyper-parameter
Randomly initialize the prediction model parameters $\theta$ and the
discriminator parameters $\theta_d$
for $iter = 1, 2, \ldots$ do
  $\Theta \leftarrow \theta, \Theta_d \leftarrow \theta_d$;
  for all $i = 1, 2, ..., k$ do
    Select a user $u$ and corresponding task $\mathcal{T}$;
    Sample a mini-batch $\mathcal{D}$ from $\mathcal{T}$;
    Update $\theta, \theta_d$ by SGD according to Eq. (10) with learning
    rate $\alpha$;
  end for
  Update the prediction model: $\theta \leftarrow \Theta + \beta(\Theta - \theta)$;
  Update the discriminator: $\theta_d \leftarrow \Theta_d + \beta(\Theta_d - \theta_d)$;
end for
```

4 EXPERIMENTS

In this section, we conduct experiments on real-world datasets to evaluate the proposed method. We aim to answer the following research questions:

RQ 1 How effective is the proposed method compared with the state-of-the-art competitors?

RQ 2 Does the proposed method actually improve the performance of tail users?

RQ 3 How do the gradient alignment and adversarial training affect the performance of the proposed method?

4.1 Experimental Setting

4.1.1 Dataset. We conduct experiments on the following publicly accessible datasets: Amazon[1], MovieLens[2] and MovieTweetings[3].

Amazon. This dataset contains a series of product purchase histories crawled from the Amazon website. Top-level product categories are used to split the dataset into separate subsets. Here, we conduct experiments on the Digital Music (**Music**) and Video Games (**Game**) category.

MovieLens. This movie rating dataset is widely used in recommendation tasks. In our experiment, we use the version that includes 1 million user ratings: MovieLens 1M (**ML1M**).

MovieTweetings. This is a dataset consisting of ratings on movies that are contained in well-structured tweets on Twitter.

For dataset preprocessing, we follow the common practice in previous works [6, 33, 36]. For all datasets, we convert explicit ratings into implicit binary feedbacks. After that, we group interactions by user ID, and construct each user's sequence by sorting according to the timestamps. In order to reduce the impact of noise data, we filter out inactive items with fewer than 5 related records, and then remove users with less than 10 feedbacks. Detail statistics of the datasets are summarized in table 2. Finally, each dataset is randomly divided into two parts according to users: 80% of the users as existing users, and the remaining 20% as new users for the analysis of cold start problem. For existing users, we use the most recent item in the interaction sequence of each user for evaluation, and the rest items for training.

4.1.2 Evaluation Metrics. The following metrics are used to evaluate the quality of recommendation, which are also widely used in previous works [9, 33, 36].

HR@N (Hit Ratio) is widely used as a measure of predictive accuracy. It represents the proportion of the desired item amongst the top-N items in all test cases. It is computed as:

$$\text{HR@}N = \frac{1}{|\mathbf{U}|}\sum_{u\in\mathbf{U}}\mathbb{I}(R_{u,g_u} \leq N), \tag{11}$$

where $\mathbb{I}$ is an indicator function. g_u is the test item of user u, and R_{u,g_u} is the rank of g_u generated by the model.

NDCG@N (Normalized Discounted Cumulative Gain) records the position of the hit by assigning larger scores on higher ranks. It is computed as:

$$\text{NDCG@}N = \frac{1}{|\mathbf{U}|}\sum_{u\in\mathbf{U}}\frac{\mathbb{I}(R_{u,g_u} \leq N)}{log_2(R_{u,g_u}+1)}. \tag{12}$$

Basically, the higher these metrics, the better the performance of the model. To make results more stable, we repeate each experiment ten times for each metric and compute the average results.

4.1.3 Baselines. To demonstrate the effectiveness of the proposed method, we compare to the following sequential user behavior modeling methods.

- **GRU4REC.** [8] This is a representative deep learning based method which adopts RNN to model users' sequential behaviors.

[1]http://jmcauley.ucsd.edu/data/amazon/
[2]http://grouplens.org/datasets/movielens/
[3]https://github.com/sidooms/MovieTweetings

Table 2: Statistics of the evaluation datasets.

Dataset	#Users	#Items	#Interactions	#Records/user	#Records/item	#Density
Music	2,831	13,410	63,054	22.27	4.70	0.163 %
Game	7,519	19,977	145,520	19.35	7.28	0.097 %
ML1M	6,040	3,706	1,000,209	165.57	269.89	4.468 %
MovieTweetings	12,605	10,910	654,658	51.94	60.01	0.476 %

- **CASER.** [26] This CNN based method attempts to capture sequential patterns on both individual-level and union-level with the help of horizontal and vertical convolutional filters.
- **SASR.** [9] This is a self-attention based sequential model which adopts an attention mechanism to identify relevant items for predicting the next item.

The above three baselines are representative sequential user behavior modeling methods based on RNN, CNN and self-attention, respectively. Since our framework is model agnostic, we use it to train the above three baselines. For fair comparisons, models trained by our method shares the same model architecture as the baselines. For simplicity, we use "-TP" denote models trained by our method compared with baselines. In the experiments, we adopt grid search to select the best parameters for each model. The embedding size is turned from $\{10, 15, \cdots, 50\}$, the regularization parameter and learning rate are selected from $\{1e^{-4}, 1e^{-3}, \cdots, 1\}$. All other hyper-parameters and initialization strategies are those suggested by the methods' authors. For our method, we adopt SGD optimizer and fix the update times $k = 2$ in inner update phase. For outer update phase, we employ adam optimizer, whose learning rate β is adam's suggested setting 0.001. For adversarial training, we simply set 20% users with the most interactions as head users and the rest as tail users, which is the same as the pareto principle [22]. The adversarial hyper-parameter λ is tuned from $\{0, 0.01, 0.1, 1, 10\}$.

4.2 Performance Comparison

To answer **RQ1**, we evaluate the recommendation performance of the proposed method and all baselines. Results are shown in Table 3 and we can draw following observations.

Observations about our framework. First, for existing users, models trained using the proposed method consistently achieve better performance than their conventional counterparts, showing the benefits of learning transferrable parameters. Our method encourages knowledge transfer across users, leading to a general model. Such a model will not be biased to either head or tail users but achieves good performance on both of them. Second, all methods perform worse on new users than existing users, which indicates that these sequential models cannot handle user cold-start problem well. Generally speaking, sequential models can make recommendations based on previous interactions without relying on the user profile. However, sequential patterns learned from existing users may not suitable for new users, which leads to performance degradation for new users. Third, models trained by our method outperform their conventional counterparts when facing new users. This result indicates that the proposed method can effectively enhance the sequential model's ability to deal with the cold start problem of new users. The reason is that our method focuses on those more diverse and noisy tail users, which is helpful for learning a more robust model that can perform well on new users. Finally, besides the above evaluation of different methods, Figure 4 shows models trained by our method are stably superior to their conventional counterparts with different lengths of the recommendation list.

Other observations. First, all methods achieve better results on MovieTweetings and ML1M datasets than Music and Game datasets. The major reason is that Music and Game datasets are more sparse than MovieTweetings and ML1M datasets, and the data sparsity declines the recommendation performance. Second, SASR outperforms than GRU4REC and CASER in most cases. The main reason is that SASR adopts the self-attention mechanism to attend items adaptively that would better reflect the user's preference. Finally, the best performing models are not consistent on different datasets, which suggests that we should choose the appropriate model according to the actual situation. It should be emphasized that our method is model agnostic and can be well adapted to various well-established sequential models.

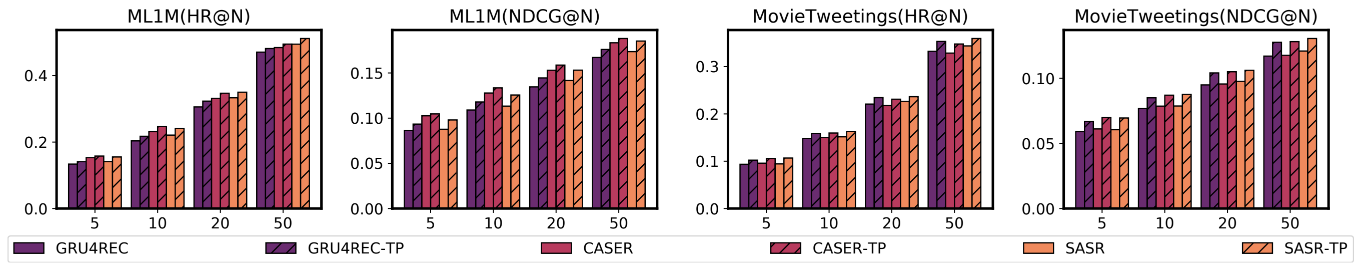
4.3 Performance on Head Users and Tail Users

To answer RQ2, we divide the evaluation by the number of user's historical interactions. Specifically, for both existing and new users, we treat 20% of the users with the most historical interactions as head users and denote the rest as tail users, which is the same as the training phase. The results are shown in Figure 5.

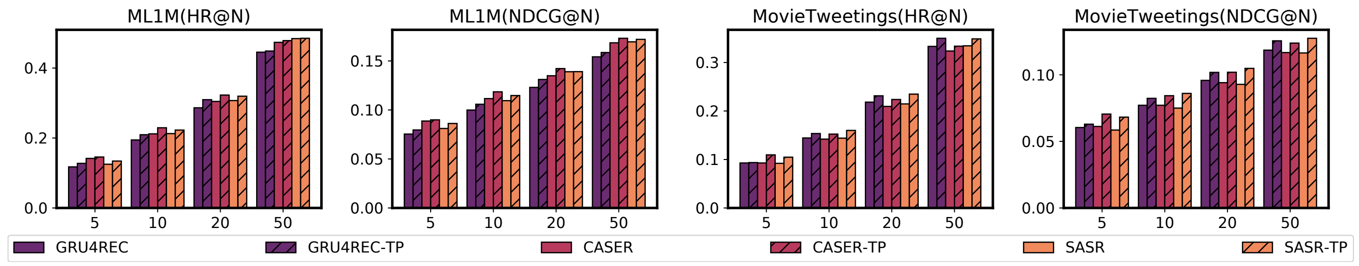
First of all, we can notice that all methods perform better on head users than on tail users. This result confirms our conjecture that head users have more data available for training than tail users, resulting in better performance. Second, for both existing and new users, models trained by our method achieve significant improvements on tail users than their conventional counterparts. On the other hand, the performance improvement of head users brought by our method is relatively small compared to tail users. Even in some cases, there is a certain decrease in the performance of head users (i.e., CASER-TP compared to CASER on both Game and MovieTweetings datasets for new users). In other words, models trained using our method will pay more attention to tail users, which helps achieve a balanced performance instead of biased towards head users. This result demonstrates that the proposed method can effectively transfer knowledge from head users to tail users, thereby achieving a general model that performs well on both head users and tail users. Finally, by comparing Game and MovieTweetings datasets, we find that the performance gap between head users and tail users on Game dataset is larger than MovieTweetings dataset. Moreover, models trained using our method can achieve a more balanced performance on Game dataset than MovieTweetings dataset, and the performance improvement of tail users is more significant. The reason could be that Game dataset is more sparse and the information gap between head users and tail users is larger than MovieTweetings dataset. In contrast, MovieTweetings dataset is relatively dense. Tail users have more data, enough to achieve good

Table 3: Performance comparison of all methods in terms of HR@10 and NDCG@10. The best results are boldfaced.

Type	Method	Music		Game		ML1M		MovieTweetings	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Existing users	GRU4REC	0.0238	0.0125	0.0324	0.0167	0.2036	0.1090	0.1482	0.0767
	GRU4REC-TP	**0.0287**	**0.0139**	**0.0359**	**0.0189**	**0.2194**	**0.1196**	**0.1586**	**0.0850**
	CASER	0.0499	0.0254	0.0598	0.0319	0.2314	0.1278	0.1502	0.0786
	CASER-TP	**0.0556**	**0.0314**	**0.0638**	**0.0330**	**0.2469**	**0.1335**	**0.1596**	**0.0870**
	SASR	0.0530	0.0295	0.0607	0.0298	0.2210	0.1133	0.1516	0.0787
	SASR-TP	**0.0605**	**0.0320**	**0.0640**	**0.0321**	**0.2411**	**0.1256**	**0.1629**	**0.0876**
New users	GRU4REC	0.0194	0.0102	0.0266	0.0135	0.1945	0.0999	0.1444	0.0771
	GRU4REC-TP	**0.0247**	**0.0111**	**0.0306**	**0.0156**	**0.2094**	**0.1058**	**0.1535**	**0.0823**
	CASER	0.0442	0.0230	0.0532	0.0259	0.2119	0.1116	0.1420	0.0771
	CASER-TP	**0.0512**	**0.0291**	**0.0552**	**0.0280**	**0.2293**	**0.1185**	**0.1523**	**0.0843**
	SASR	0.0495	0.0242	0.0532	0.0259	0.2127	0.1083	0.1440	0.0750
	SASR-TP	**0.0548**	**0.0295**	**0.0566**	**0.0282**	**0.2227**	**0.1147**	**0.1599**	**0.0860**



(a) Recommendation for existing users



(b) Recommendation for new users

Figure 4: Top-N recommendation evaluation with different values of N on HR and NDCG.

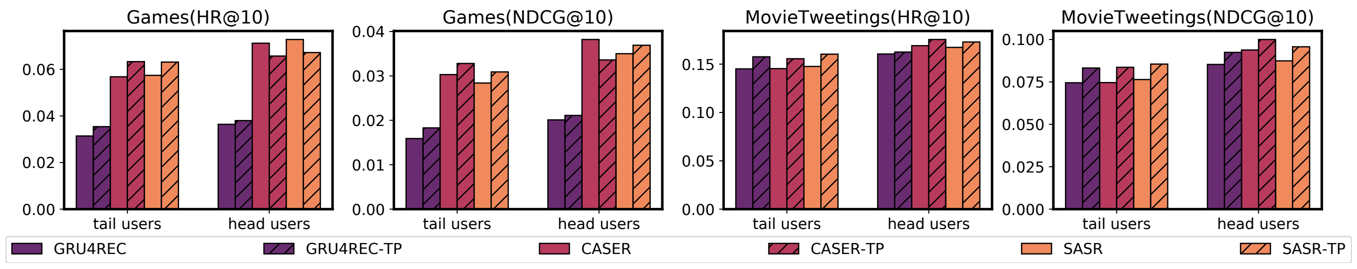
performance. Therefore, the performance improvement brought by learning transferrable parameters of our method is weaker on MovieTweetings dataset than Game dataset.

4.4 Impact of Gradient Alignment and Adversarial Training

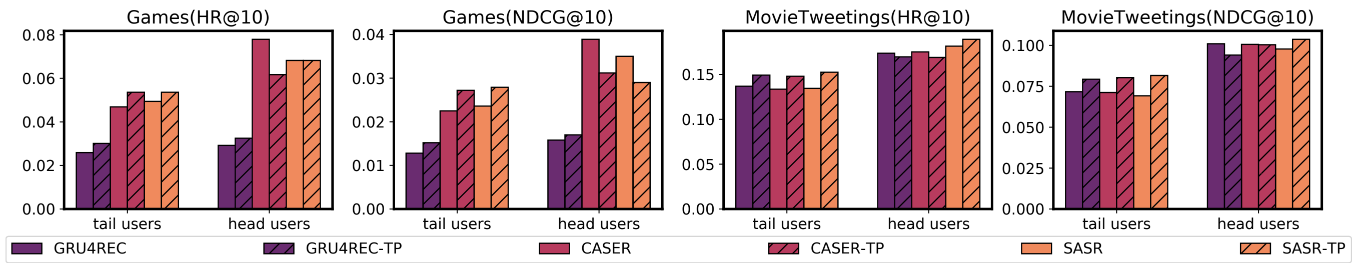
For the proposed method, there are two key parameters related to gradient alignment and adversarial training. The first is the inner update times k. Figure 6 shows the performance of our method for varying inner update times k on ML1M and MovieTweetings datasets. The second is the adversarial hyper-parameter λ. Figure 7 shows the performance of our method for varying adversarial hyper-parameter λ on ML1M and MovieTweetings datasets.

First, by examining the influence of the inner update times k, we found that the performance of our method is worst when $k = 1$. This is reasonable since it only optimizes the expected loss over all users, without considering the gradient alignment between different users, which can learn transferrable parameters from an optimization perspective. By further investigating the performance when $k \geq 2$, we found that there are only slight differences when increasing the inner update times compared to the significant improvement as k from 1 to 2. One possible explanation is that knowledge transfer across users can be achieved for any value of $k \geq 2$. Increasing the value of k, which is greater than 2 cannot significantly change the effect of knowledge transfer.

Second, by examining the influence of λ, we found that better performance is achieved by balancing the impact of prediction loss and adversarial loss, while either a large or small value of λ will adversely degrade the performance. Presumably, this is because a too large value of λ means the prediction model spends too much



(a) Recommendation for existing users



(b) Recommendation for new users

Figure 5: Recommendation evaluation on head users and tail users.

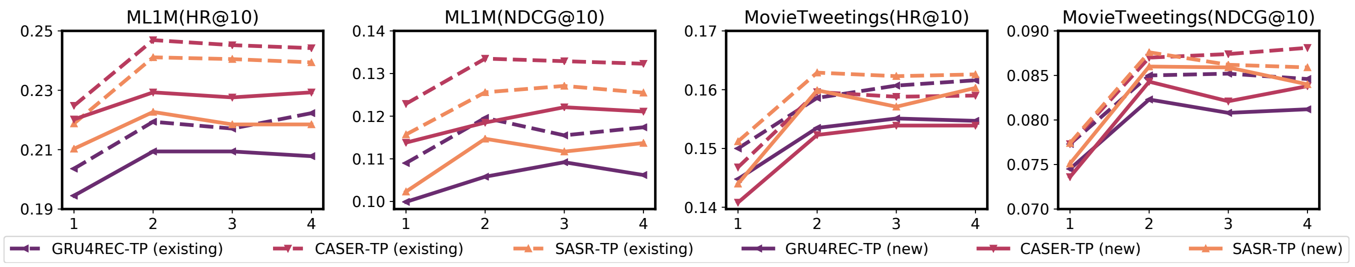


Figure 6: Impact of inner update times k.

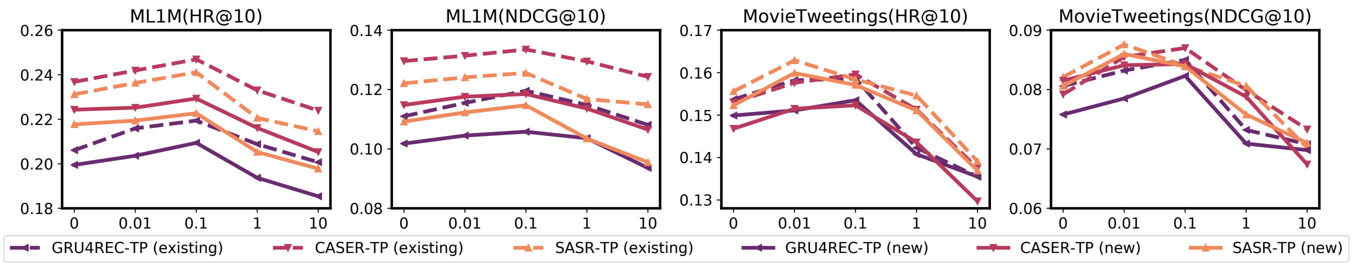


Figure 7: Impact of adversarial parameter λ.

effort to learn frequency-agnostic sequential embedding, resulting in insufficient sequential modeling problem. On the contrary, a too small value of λ cannot effectively transfer knowledge from head users to tail users, resulting in a biased model, then reducing the performance of our method.

5 CONCLUSION

In this paper, we propose to solve the long-tailed distribution problem in sequential user behavior modeling by learning transferrable parameters. Specifically, we propose a gradient alignment optimizer to encourage knowledge transfer from the optimization perspective. Moreover, we introduce an adversarial training method to learn frequency-agnostic sequential embedding, which facilitates knowledge transfer from the feature perspective. Experiments on real-world datasets demonstrate the effectiveness of the proposed method, by comparing with the state-of-the-art baselines.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation Singapore, and AI Singapore under its Research Programme (AISG-RP-2018-001).

REFERENCES

[1] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond globally optimal: Focused learning for improved recommendations. In *Proceedings of the 26th International Conference on World Wide Web*. 203–212.
[2] Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. 2018. Transferring knowledge across learning processes. *arXiv preprint arXiv:1812.01054* (2018).
[3] Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In *Advances in neural information processing systems*. 1334–1345.
[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[6] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.
[7] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 549–558.
[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[9] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.
[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[11] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.
[12] Chenghao Liu, Steven CH Hoi, Peilin Zhao, Jianling Sun, and Ee-Peng Lim. 2016. Online adaptive passive-aggressive methods for non-negative matrix factorization and its applications. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1161–1170.
[13] Chenghao Liu, Tao Jin, Steven CH Hoi, Peilin Zhao, and Jianling Sun. 2017. Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Machine Learning* 106, 5 (2017), 651–670.
[14] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 825–833.
[15] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
[16] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).
[17] James R Norris. 1998. *Markov chains*. Number 2. Cambridge university press.
[18] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.
[19] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.
[20] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2018. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910* (2018).
[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in neural information processing systems*. 2234–2242.
[22] Robert Sanders. 1987. The Pareto principle: its use and abuse. *Journal of Services Marketing* (1987).
[23] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.
[24] Mohit Sharma and George Karypis. 2019. Adaptive matrix completion for the users and the items in tail. In *The World Wide Web Conference*. 3223–3229.
[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[26] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 565–573.
[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[28] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 403–412.
[29] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[30] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2017. ST-SAGE: A spatial-temporal sparse additive generative model for spatial item recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 3 (2017), 1–25.
[31] Weiqing Wang, Hongzhi Yin, Xingzhong Du, Quoc Viet Hung Nguyen, and Xiaofang Zhou. 2018. Tpm: A temporal personalized model for spatial item recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 6 (2018), 1–25.
[32] Jianwen Yin, Chenghao Liu, Jundong Li, BingTian Dai, Yun-chen Chen, Min Wu, and Jianling Sun. 2019. Online Collaborative Filtering with Implicit Feedback. In *International Conference on Database Systems for Advanced Applications*. Springer, 433–448.
[33] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order Attentive Ranking Model for Sequential Recommendation. (2019).
[34] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2018. A Simple but Hard-to-Beat Baseline for Session-based Recommendations. *arXiv preprint arXiv:1808.05163* (2018).
[35] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. 2019. Lookahead Optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*. 9593–9604.
[36] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. 2019. Next Item Recommendation with Self-Attentive Metric Learning. (2019).
[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.
[38] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 580–588.