FairSR: Fairness-aware Sequential Recommendation through Multi-Task Learning with Preference Graph Embeddings

CHENG-TE LI and CHENG HSU, National Cheng Kung University
YANG ZHANG, CISPA Helmholtz Center for Information Security

Sequential recommendation (SR) learns from the temporal dynamics of user-item interactions to predict the next ones. Fairness-aware recommendation mitigates a variety of algorithmic biases in the learning of user preferences. This article aims at bringing a marriage between SR and algorithmic fairness. We propose a novel fairness-aware sequential recommendation task, in which a new metric, *interaction fairness*, is defined to estimate how recommended items are fairly interacted by users with different protected attribute groups. We propose a multi-task learning-based deep end-to-end model, FairSR, which consists of two parts. One is to learn and distill personalized sequential features from the given user and her item sequence for SR. The other is fairness-aware preference graph embedding (FPGE). The aim of FPGE is two-fold: incorporating the knowledge of users' and items' attributes and their correlation into entity representations, and alleviating the unfair distributions of user attributes on items. Extensive experiments conducted on three datasets show FairSR can outperform state-of-the-art SR models in recommendation performance. In addition, the recommended items by FairSR also exhibit promising interaction fairness.

CCS Concepts: • **Information systems** → **Data mining**; **Retrieval models and ranking;**

Additional Key Words and Phrases: Fairness-aware models, sequential recommendation, knowledge graph embedding, multi-task learning

ACM Reference format:
Cheng-Te Li, Cheng Hsu, and Yang Zhang. 2022. FairSR: Fairness-aware Sequential Recommendation through Multi-Task Learning with Preference Graph Embeddings. *ACM Trans. Intell. Syst. Technol.* 13, 1, Article 16 (February 2022), 21 pages.
https://doi.org/10.1145/3495163

1 INTRODUCTION

Sequential recommendation (**SR**) is a crucial research task in **recommender systems** (**RS**) [46]. SR considers the chronological order of user-item interactions, and models how users' recent successively accessed items affect the choices of the next ones. To be specific, given a recent sequence

This work is supported by Ministry of Science and Technology (MOST) of Taiwan under grants 110-2221-E-006-136-MY3, 110-2221-E-006-001 and 110-2634-F-002-051.
Authors' addresses: C.-T. Li (corresponding author) and C. Hsu, National Cheng Kung University, No. 1, University Rd., East District, Tainan City 701, Taiwan; emails: chengte@mail.ncku.edu.tw, hsuchengmath@gmail.com; Y. Zhang, CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, Saarbrücken 66123, Germany; email: zhang@cispa.de.

2157-6904/2022/02-ART16 $15.00
https://doi.org/10.1145/3495163

of items interacted by a user, SR aims at learning from the sequence to find which items will be interacted by her in the near future. The SR task differs from conventional RS. While RS tends to capture the global user preferences on items [13, 37], SR imposes the sequential dynamics of user-item interactions. Hence, SR requires the learning of long-term and short-term interests and intents of users [22, 30] in predicting the next items.

On the other hand, while algorithmic fairness is getting attention in the machine-learning community [24], fairness in recommender systems is investigated in various aspects. Typical issues on recommendation fairness can be mainly divided into three groups. The first is dealing with *popularity bias* concerning that few popular items are over-represented in the models [2, 9]. The second is tacking *demographic bias*, in which the representations of users with imbalanced attributes (e.g., gender and age) cannot be equally learned and lead to differentiated and unfair performance [9, 10, 28]. The third is imposing *statistical parity* into recommendation, which aims at ensuring similar probability distributions of item ratings for those users in different protected attribute groups [42, 49].

Despite existing methods on fairness-aware recommendation receive satisfactory results, we think there remain several opportunities and challenges. First, to the best of our knowledge, none of the existing studies targets at considering fairness into sequential recommendation. This work is an essential attempt to define and solve a fairness-aware SR problem. Second, previous fairness-aware recommendations discussed above are not aware of the *filter bubble* effect [26], which states that online personalization tends to effectively isolate users from a diversity of viewpoints. Since people prefer to interact with what they liked or interacted with before, learning-based recommenders will reinforce user preferences to satisfy them [25]. We regard it as a kind of unfair recommendation for users who want to pursue novel or diverse items. Therefore, it is worthwhile to design a new fairness concept so that recommenders can follow to generate items against the filter bubble. Third, **knowledge graphs** (**KG**) that connect items based on their metadata or attributes had been proven to be promising for recommendation systems [6, 34, 36]. Yet KG is not explored for a sequential recommendation. Besides, while existing KG models item-item relationships, user attributes can be incorporated into KG so that the item embeddings can encode the knowledge on user traits.

In this article, we propose a novel **fairness-aware sequential recommendation** (**FairSR**) task. Given user-attribute groups to be protected, i.e., a set of attribute groups that are concerned for fairness (e.g., "Female & Age 20–29" and "Male & Age 50–59"), and the recent item subsequence of a user, we aim at not only accurately predicting the next items, but also require that the recommended items lead to ***Interaction Fairness*** (**IF**). Better IF means that recommended items tend to be equally interacted by users of different protected attribute groups. Taking Figure 1 as an example, we assume two protected attribute groups 1 and 2 are specified. Given sequences of five items for users M and F, the recommended items of F are fairer in terms of IF, while those of M are unfair. The reason is each recommended item of F tends to be equally interacted by groups 1 and 2, but it is not for M. Note that although the given item sequences of users are not fair, we still require their recommended next ones to be fair in terms of IF. When fighting against the filter bubbles, online social media platforms are in need of interaction fairness for their recommender systems. If an SR system is aware of interaction fairness, a user will have a high possibility to receive items being interacted by other users with diverse attribute groups, and the effect of filter bubbles can be mitigated.

As aforementioned, existing studies have investigated three common types of unfairness issues in the recommendation, including popularity bias, demographic bias, and statistical parity. The proposed IF in SR is to mitigate the filter bubble effect in recommender systems, rather than solving the three types of unfair recommendation. The difference between IF and the three types of unfairness is two-fold. First, IF considers how other users interact with the recommended items,

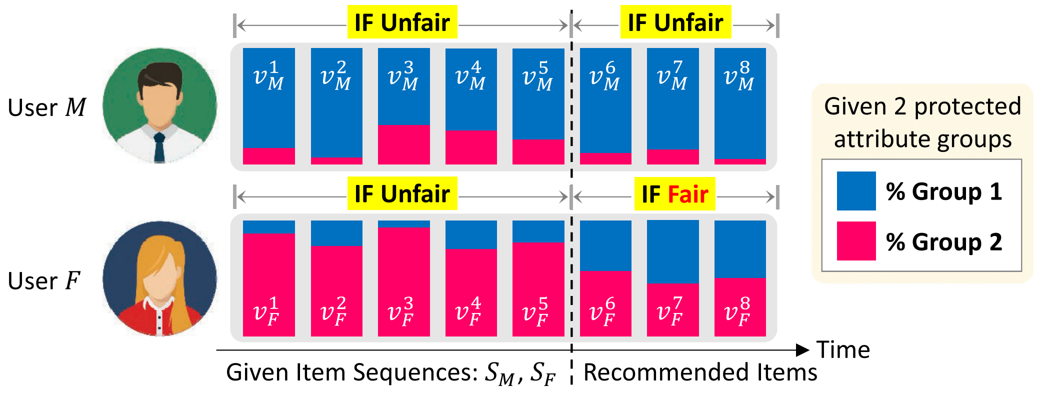

IF Unfair
IF Unfair
IF Unfair
IF Fair
User M
User F
v_M^1 v_M^2 v_M^3 v_M^4 v_M^5 v_M^6 v_M^7 v_M^8
v_F^1 v_F^2 v_F^3 v_F^4 v_F^5 v_F^6 v_F^7 v_F^8
Given 2 protected attribute groups
% Group 1
% Group 2
Time
Given Item Sequences: S_M, S_F
Recommended Items

Fig. 1. Elaboration of fairness-aware SR. Note that for each user, every colored bar is an item, and their associated notations v_M^i and v_M^i indicate the i th item that user M and F interact with. The Blue and red parts inside each bar exhibit the proportions of historical interactions with an item created by users belonging to attribute groups 1 and 2, respectively.

which are expected to be fairly treated by user groups with different protected attributes. Both popularity bias and demographic bias concern about users/items with less attention (i.e., less popular items, the minority attribute of users), and the proposed methods want them to be fairly treated in the construction of recommender systems. Second, the recommended items with higher IF values provide higher diversity for users, but those with maintaining statistical parity is to keep the distribution of item ratings the same before and after recommendation, instead of dealing with the issue of diversity.

We propose a **multi-task learning** (**MTL**)-based end-to-end deep model, `FairSR`, to solve the fairness-aware sequential recommendation task. The main task aims at performing SR by learning sequential features from the given item sequence. A personalized feature gating, as well as two convolution mechanisms, are performed to produce effective sequence representations that encode user preferences and sequential patterns. Another task of FairSR is to learn **fairness-aware preference graph embeddings** (**FPGE**). Borrowing from the idea of knowledge graph embeddings [35], we encode both attributes of users and items and their relations into entity embeddings. While user-item interactions are biased to some attribute groups, we propose a fairness-aware triplet sampling to generate positive triplets of the head, relation, and tail so that the bias can be mitigated in FPGE. Two tasks are connected with each other through a **cross item-preference learning** (**CIPL**), which encodes shared features between items in SR and their corresponding entities in FPGE.

We summarize the contribution of this article as follows:

— We propose a novel fairness-aware sequential recommendation problem that brings a marriage between SR and recommendation fairness. We define a new fairness metric, interaction fairness, which quantifies the degree of the filter bubble effect by estimating how recommended items are interacted by protected attribute groups.
— We develop a new multi-task learning-based deep model, `FairSR`,[1] to solve the problem. The main task is SR that predicts the next items based on the learned personalized sequential features. The other task is FPGE that encodes both knowledge and fairness of attributes into entity embeddings.

[1]The FairSR code and datasets can be accessed via this link: https://github.com/fairsr/fairsr.

— Extensive experiments conducted on three real datasets exhibit not only promising recommendation performance of FairSR, compared to several state-of-the-art models, but also a fair recommendation in terms of interaction fairness.

We organize this article as follows. Section 2 reviews relevant studies, followed by Section 3 that describes the problem statement. In Section 4, we present the technical details of the proposed FairSR model. We give the experimental results in Section 5, and conclude this work in Section 6.

2 RELATED WORK

The relevant studies can be categorized into three groups, SR models, fairness-aware RS, and KG enhanced recommender systems.

SR Models. In deep SR models, recurrent neural networks [14, 15, 41] and convolutional neural networks [30] are used to extract long-term and short-term sequential features. SASRec [18] is a self-attentive model that can identify the most significant items for prediction. HGN [22] is a hierarchical gating neural network that adopts feature gating and instance gating to determine what item features should be used for recommendation. MARank [44] is a multi-order attentive ranking model that unifies both individual- and union-level item-item interaction into the preference inference model from multiple views. NextItNet [45] is a dilated convolution-based generative method to learn long-range dependencies in the item sequence. JODIE [20] is a coupled recurrent neural network model that jointly learns the embedding trajectories of users and items from a sequence of temporal interactions. Recent studies on SR deal with various limitations in real-world scenarios. RetaGNN [16] delivers a graph neural network-based holistic sequential recommendation model that accommodates SR under conventional, inductive, and transferable settings. Mecos [48] concentrates on mitigating the item cold-start problem in SR without utilizing side information. A category-aware collaborative SR model [5] further proposes to improve the performance by utilizing multi-grained category information of items.

Fairness-aware RS. Various kinds of fairness are investigated and developed for RS. The most well-known three are popularity bias [2, 9], demographic bias [9, 10, 28], and statistical parity: [42, 49]. Popularity bias means a few popular items are over-represented in the models. A personalized diversification re-ranking approach is developed to increase the representations of unpopular items [1]. Demographic bias indicates that the representations of users with imbalanced attributes (e.g., gender and age) cannot be equally learned and lead to differentiated and unfair performance. Incorporating multiple-source data [10] and performing data augmentation [28] can alleviate demographic bias. Statistical parity aims at ensuring similar probability distributions of item ratings for users in different protected attribute groups. Extracting and isolating sensitive information from the factorized matrices [49] and re-ranking recommended items based on required attribute distributions [12] can improve statistical parity. In addition to such three fairness issues, Jiang et al. [17] deal with account-level recommendation bias by identifying users. Although fairness-aware embedding learning methods [4, 27] fairly encode attributes for node representation learning in graphs, they do not target at sequential recommendation. Beutel et al. [3] devise a pairwise regularizer to improve pairwise fairness metrics in RS. Rather than considering fairness in RS, our work aims at imposing fairness into SR.

KG-enhanced RS. KG embedding [35] provides additional features depicting the association between items through metadata and attributes. KGs are leveraged in various ways, including propagating user preferences over knowledge entities by RippleNet [33], multi-task learning with KG Embedding by MKR [34], applying graph attention on a user-item-attribute graph by KGAT [36], adopting LSTM to model sequential dependencies of entities and relations [39], and integrating induction of explainable rules from KG by RuleRec [23]. MARINE [11] combines homogeneous and heterogeneous graph embedding learning mechanisms to recommend links between entities. Furthermore, KGPL [31] assigns pseudo-positive labels to unobserved samples through knowledge graph neural network-based predictions so that the recommendation model can better deal with the cold-start issues. KGPolicy [40] leverages rich relations between items in the knowledge graphs to sample high-quality negatives and boost the performance of recommenders. JNSKR [7] presents a non-sampling and efficient learning mechanism based on an attentive neural network for better knowledge graph-enhanced recommenders. KGIN [38] learns user intents by modeling attentive combinations of relations in the knowledge graph to enhance the recommendation performance and bring model interpretability. Although these studies successfully apply knowledge graphs with various mechanisms for better recommender systems, exploiting KG to assist sequential recommendation is not well explored yet.

The main difference between our work and existing studies consists of three parts. First, in the task of sequential recommendation, none of the existing methods (e.g., [18, 22, 30, 33, 34]) can simultaneously model the personalized sequential features for users and the latent correlation behind items via the concept of knowledge graphs. We believe that item representation learning by utilizing the relations between items and attributes can better model user preferences. Second, while some of the conventional recommendation models are devised to mitigate various unfairness issues (e.g., FATR [49] for statistical parity, Fair-PSL [9] for demographic bias, and miscalibration [2] for popularity bias), existing SR methods cannot deal with any fairness issues. The proposed Fairness-aware Triplet Sampling mechanism in FairSR brings the item embeddings to be aware of interaction fairness. Third, in FariSR, we devise the Personalized Feature Gating component, which originates from gated recurrent unit [8] in the task of language modeling, to adaptively select items specialized to the targeted user.

3 PROBLEM STATEMENT

In typical recommendation, we have a set of M users $\mathcal{U} = \{u_1, \dots, u_M\}$ and a set of N items $\mathcal{V} = \{v_1, \dots, v_N\}$. The matrix of user-item interactions is denoted by $\mathbf{Y} \in \mathbb{R}^{M \times N}$ based on the implicit feedback from users, in which $y_{uv} = 1$ indicates user u had ever interacted with item v; otherwise, $y_{uv} = 0$. While a user u sequentially interact with different items in a chronological manner, we denote the corresponding item sequence as $\mathcal{S}^u = (s_1^u, s_2^u, \dots, s_L^u)$, where $L = |\mathcal{S}^u|$ and $s_i^u \in V$ is an item index that user u has interacted with.

Let A be the universal set of demographic attribute groups. A demographic attribute group $a \in A$ can be the gender (g), age (o), country (c), or their combinations. We denote the set of all possible values of a as Z^a, and denote a specific value as $z^a \in Z^a$. Let the function $\mathcal{A}^a : \mathcal{U} \rightarrow Z^a$ map a user to her attribute groups. An attribute group value $\mathcal{A}^a(u)$ of user u can be, for example, "male" or "female" for attribute group {Gender}, and "10–19 & US", "20–29 & UK" or "30–39 & JP" for attribute group {Age, Country}. To enhance the readability, we use *attribute* and *attribute value* to represent *demographic attribute group* and *demographic attribute group value*, respectively, throughout the article.

We consider that fair recommendation expects an *unbiased* distribution on recommended items with respect to a certain attribute. Hence, a criterion is required to measure the *equality* of user interactions with items. Given an attribute value z^a of attribute $a \in A$, we first define *adoption proportion* with respect to a certain item. The adoption proportion measures the percentage of users with a specific attribute value adopting an item among all users who had ever interacted with the item. Higher values of adoption proportion (e.g., ≈ 1.0) indicate that the item is highly biased to a specific attribute value, and thus tends to be unfair with respect to the attribute. The definition of adoption proportion is given by

$$p_v(z^a) = \frac{|\{u|\mathcal{A}^a(u) = z^a, u \in \mathcal{U}(v)\}|}{|\mathcal{U}(v)|}, \tag{1}$$

where $\mathcal{U}(v)$ is the set of users who had ever interacted with item v.

Since an item can be adopted by users with different values of a particular attribute, we further define the *adoption equality* to collectively quantify the degree that an item is equally interacted based on adoption proportion using all values of an attribute. The adoption equality considers all possible values of an attribute to measure the fairness of an item being interacted. If each value of an attribute contributes nearly the same to an item in terms of adoption proportion, the adoption equality gets a higher score, and we can say that this item tends to be fairly interacted by users with different values of that attribute. That said, the adoption of the item is not biased to users with specific attribute values. Then we define *adoption equality* of item v, denoted by $e^a(v)$, based on information entropy, given by

$$e^a(v) = -\sum_{z^a \in Z^a} p_v(z^a) \log p_v(z^a). \tag{2}$$

A higher value of $e^a(v)$ indicates higher adoption equality of item v.

Given an attribute $a \in A$, for a list of items $\mathcal{S}^u_{t:L}$ $(t < L)$ that are recommended to user u, we define its corresponding IF score $\mathcal{F}^a(u)$ with respect to attribute a based on adoption equality, given by

$$\mathcal{F}^a(u) = \sum_{v \in \mathcal{S}^u_{t:L}} e^a(v). \tag{3}$$

A higher value of $\mathcal{F}^a(u)$ indicates that a recommendation algorithm tends to recommend items to user u with a higher fairness degree for attribute a. That said, the recommended items tend to be unbiased to a specific attribute value $z^a \in Z^a$.

The FairSR problem can be defined as below. Given the earlier subsequence $\mathcal{S}^u_{1:t}$ $(t < L)$ of every user $u \in \mathcal{U}$, we aim at recommending a list of items from item set $\mathcal{V}$ to each user, i.e., predict whether user u will interact with item $v \in \mathcal{V}$ after time t (whether the items in ground truth $\mathcal{S}^u_{t:L}$ will appear in the recommended list), and maximize the corresponding interaction fairness $\mathcal{F}^a(u)$.

4 THE PROPOSED FAIRSR MODEL

The overview of our FairSR framework is presented in Figure 2, which consists of two tasks. One is the main task SR, and the other is FPGE shown in Figure 3. When an item subsequence is entered, the CIPL is used to learn shared features between items and their corresponding entities. Then for the SR part, we design a personalized feature gating to distill effective sequential features, followed by horizontal and vertical convolution mechanisms to extract sequential patterns. By combining sequential features with item-item correlation and global user-item latent factors, SR makes the prediction. For the FPGE part, we first construct a preference graph that depicts the relations between items and attributes. We devise a relational attention-based information passing mechanism to learn entity embeddings. The aim of FPGE is to predict tails based on embeddings of heads and relations. Our framework is trained by alternately optimizing the two tasks.

Here, we summarize the novelty of this work into four points. First, we propose a novel task, fairness-aware sequential recommendation, which aims at accurately recommending the next items and simultaneously mitigating the effect of filter bubble in online services. Second, we present a new fairness measure, IF, to quantify the degree of the filter bubble effect. Our fairness-aware task is to recommend items that can increase IF values. Third, we present a novel multi-task

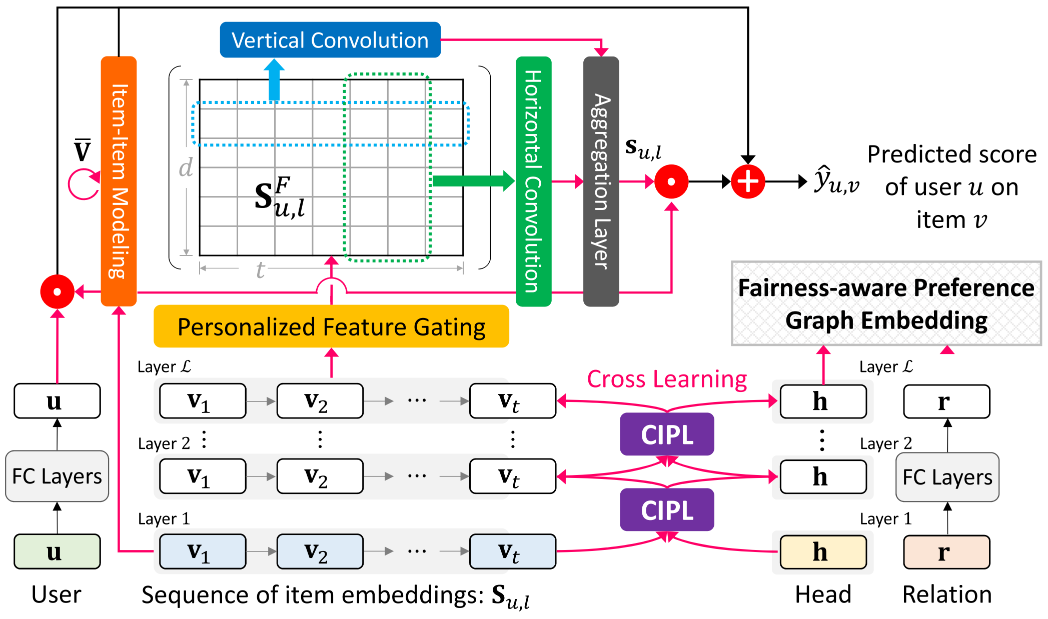


Fig. 2. The main framework of FairSR.

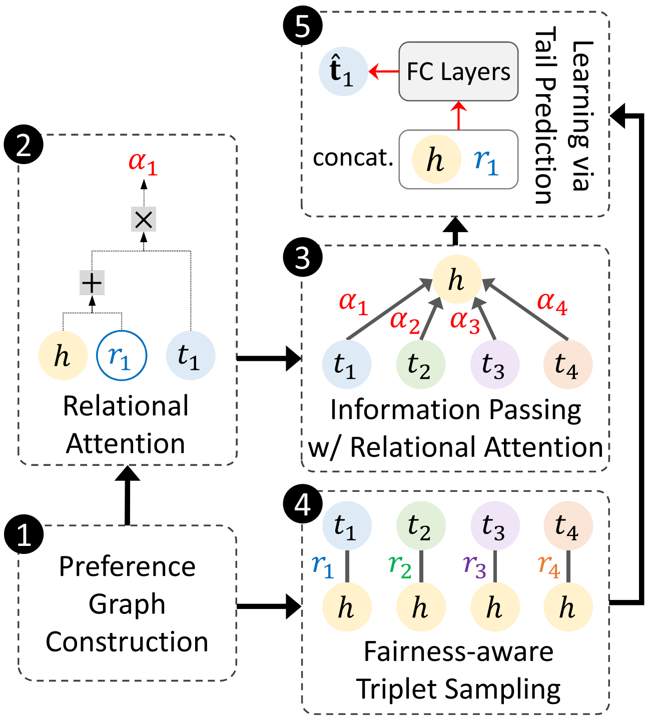


Fig. 3. Fairness-aware Preference Graph Embedding.

learning-based model, FairSR, to tackle the proposed task. FairSR brings a marriage between personalized sequential feature learning and preference graph embedding in the context of sequential recommendation. Fourth, in the component of preference graph embedding, a new fairness-aware triplet sampling strategy is proposed to ensure that the item embedding learning can encode the fairness across the protected attribute groups.

4.1 Sequential Feature Learning

The basic idea of sequential recommendation is that the recent subsequent items can to some extent influence the adoption of the next items. Hence, existing studies have presented a variety of models to learn the representations of item (sub)sequences, such as convolution neural network [30], recurrent neural network [14], self-attention mechanism [18], and feature gating method [22]. To robustly model the correlation between the current subsequence of items and the next items to be recommended, we propose a joint gating and convolutional subnetwork, which combines *personalized feature gating* and two *convolutional mechanisms*. The personalized feature gating is to select significant sequential features that are positively related to the next item prediction. Convolutional mechanisms aim at modeling sequential patterns from the perspectives of both *union*-level and *point*-level. This section consists of four phases: (a) Item Embedding Layer, (b) Personalized Feature Gating, (c) Convolutional Layers, and (d) Aggregation Layer.

User & Item Embedding Layers. The input of our model is the one-hot vectors for each user and each item. By feeding one-hot vectors into the user and item embedding layers, where each of which is a hidden layer with dimension d, each user and each item can be represented by a low-dimensional real-value dense vector. Let the user embeddings be $\mathbf{U} \in \mathbb{R}^{d \times M}$, and also let the item embeddings be $\mathbf{V} \in \mathbb{R}^{d \times N}$, where d is the embedding dimension and set as $d = 32$ by default throughout the article. Note we denote a user embedding vector and an item embedding vector as $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$, respectively. Given the l th item subsequence $\mathcal{S}^u_{1:t}$ of a certain user u, its corresponding embeddings can be represented by

$$\mathbf{S}_{u,l} = \begin{bmatrix} | & & | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_j & \cdots & \mathbf{v}_t \\ | & & | & & | \end{bmatrix},$$

where $\mathbf{S}_{u,l} \in \mathbb{R}^{d \times t}$, and $\mathbf{v}_t \in \mathbb{R}^d$ can be retrieved from the item embedding matrix $\mathbf{V}$.

Personalized Feature Gating. For different users, it is various that which of items in the current sequence are more effective in predicting next ones. The selection of representative items needs to be user-specific. In other words, the feature gating should be personalized to every user. To fulfill feature gating, we take advantage of the ***gated linear unit*** (**GLU**) [8], which is originally devised to detect and propagate effective word embeddings for predicting next word in natural language modeling. We exploit GLU to find what a user u cares about along the sequence of items $\mathbf{S}_{u,l}$. We refer to inner product-based feature gating [22] to distill item features to users' preferences. The distilled features can be derived through

$$\mathbf{S}^F_{u,l} = \mathbf{S}_{u,l} \otimes \sigma\left(\mathbf{S}_{u,l} \cdot \mathbf{W}_{g_1} + \mathbf{u} \cdot \mathbf{w}^\top_{g_2}\right), \tag{4}$$

where $\mathbf{S}^F_{u,l} \in \mathbb{R}^{d \times t}$, $\mathbf{u} \in \mathbb{R}^d$ is the embedding of user u, $\mathbf{W}_{g_1} \in \mathbb{R}^{t \times t}$, and $\mathbf{w}_{g_2} \in \mathbb{R}^t$ are learnable weights. Each user embedding $\mathbf{u}$ is generated through feeding the one-hot vector of a user into a d-dimensional hidden layer, where d is set as 32 by default. $\otimes$ is the element-wise product between matrices, and σ is the sigmoid function. The distilled sequential feature matrix $\mathbf{S}^F_{u,l}$ captures user preferences from past items to the next one, and will be used in follow-up layers.

Convolutional Layers. Equipped with the distilled sequential features, we aim at learning sequential patterns by treating $\mathbf{S}_{u,l}^F$ as an image. We exploit convolution filters to search for sequential patterns. Two convolutional filters are adopted. One is *horizontal convolution*, and the other is *vertical convolution*. Horizontal convolution filters ($h \times d$ matrices, $h = 2$ by default) are created to find *union*-level sequential patterns, which mean how features of a few consecutive items lead to a particular next item. Vertical convolution filters ($t \times 1$ matrices) are devised to learn *point*-level sequential patterns, which refer to how each feature (dimension) distributes over the item subsequence affect the prediction.

Horizontal Convolution Filters. This layer contains n_h horizontal filters $\Psi^j \in \mathbb{R}^{d \times h}$, $1 \leq j \leq n_h$, where $h \in \{1, \ldots, t\}$ is the width of a filter. For instance, in Figure 2, we can choose $n_h = 4$ filters if $t = 2$, in which two for each h in $\{1, 2\}$. Every filter Ψ^j moves from left to right on $\mathbf{S}_{u,l}^F$, and produces convolved values with horizontal dimensions from item 1 to $t - h + 1$ in item subsequence $\mathcal{S}_{1:t}^u$. The resulting vector $\mathbf{c}^j$ of horizontal convolution with filter Ψ_j can be obtained by

$$\mathbf{c}^j = \left[\cdots, \phi_c(\mathbf{S}_{i:i+h-1} \odot \Psi^j), \cdots\right], \tag{5}$$

where $j = 1, 2, \ldots, t-h+1$, $\mathbf{S}$ is $\mathbf{S}_{u,l}^F$ for simplicity, $\odot$ is the element-wise multiplication, and $\phi_c(\cdot)$ is the activation function for convolutional layers (*tanh* is used by default). Since we have n_h filters, we can derive the resulting matrix $\mathbf{D} \in \mathbb{R}^{(t-h+1)\times n_h} = [\mathbf{c}^1, \mathbf{c}^2, \ldots, \mathbf{c}^{n_h}]$.

Vertical Convolution Filters. This layer has n_v vertical filters $\Lambda^j \in \mathbb{R}^{1\times t}$, $1 \leq j \leq n_v$. Every filter Λ^j operates on each embedding dimension of $\mathbf{S}_{u,l}^F$, and thus generates d convolved values. Since the derivation of convolved values can be considered as the weighted sum of Λ^j on each embedding dimension of $\mathbf{S}_{u,l}^F$, we can depict the vertical convolution operation using the inner product, and have the resulting vector $\bar{\mathbf{c}}^j$ by

$$\bar{\mathbf{c}}^j = \left[\cdots, \sum_{i=1}^{t} \Lambda_i^j \cdot \mathbf{S}_i^\top, \cdots\right], \tag{6}$$

where $\mathbf{S}_i$ is the ith row of matrix $\mathbf{S}_{u,l}^F$. In other words, the vertical convolutional layer aggregates the embeddings of past t items through a variety of filters. Since we have n_v filters, we can obtain the resulting matrix $\bar{\mathbf{D}} \in \mathbb{R}^{d\times n_v} = [\bar{\mathbf{c}}^1, \bar{\mathbf{c}}^2, \ldots, \bar{\mathbf{c}}^{n_v}]$.

Aggregation Layer. Since the aim is to generate the effective representation for user u's item subsequence $\mathbf{S}_{u,l}^F$, we aggregate the convolution results into sequence-level embeddings. We apply *max pooling* to every horizontal convolved vector $\mathbf{c}^j$ and every vertical convolved vector $\bar{\mathbf{c}}^j$. By doing so, we can extract the significant features from all values produced by a particular filter. As a result, for n_h horizontal and n_v vertical filters, we can obtain the corresponding aggregated vectors $\mathbf{s}_{u,l}^{hc} \in \mathbb{R}^{n_h}$ and $\mathbf{s}_{u,l}^{vc} \in \mathbb{R}^{n_v}$. Then such two vectors are concatenated to be the final extracted sequential feature vector, denoted by $\mathbf{s}_{u,l} = [\mathbf{s}_{u,l}^{hc}, \mathbf{s}_{u,l}^{vc}]$. In short, the learned subsequence embedding $\mathbf{s}_{u,l}$ encodes not only user preferences on past items from personalized feature gating, but also captures the sequential patterns in both horizontal and vertical aspects.

4.2 Cross Item-Preference Learning

Since items can be characterized by how they are interacted by users, we model feature interactions between items and entities in the preference graph. A cross item-preference learning module is developed. We adopt the *cross&compress unit* [34] to implement the cross item-preference learning. To be specific, the CIPL module is to model high-order interactions between items and their corresponding entity features, which captures the ways that users with various attributes express their preferences on items. CIPL can automatically control the cross knowledge transfer between tasks of sequential recommendation and entity embedding learning. Through the learnable weights that bring embeddings of items and entities together in CIPL, the two tasks can affect and complement one another. The CIPL module consists of two main steps, one is *cross*, and the other is *compress*. The *cross* step is devised to produce an interaction matrix between items and entities based on their representations. That said, the interaction matrix models how both sides correlate with each other. The *compress* step utilizes the learned interaction matrix to map the embeddings of items and entities into the same space, i.e., generating the updated embeddings of items and entities at the next layer.

Each item $v \in \mathcal{V}$ is associated with an entity e in the preference graph. We build a $d \times d$ pairwise interaction embedding matrix $\mathbf{C}_l$ for the item embedding $\mathbf{v}_l \in \mathbb{R}^d$ and the entity embedding $\mathbf{e}_l \in \mathbb{R}^d$ at the lth neural network layer. The matrix $\mathbf{C}_l$ depicts the cross feature interactions between every item v and every entity e, and can be derived by $\mathbf{C}_l = \mathbf{v}_l \mathbf{e}_l^\top \in \mathbb{R}^{d \times d}$, where d is the dimension of hidden layers. Then we generate the next-layer $l+1$ feature vectors of items and entities by projecting the cross feature matrix into their embedding space, given by

$$\begin{aligned} \mathbf{v}_{l+1} &= \mathbf{C}_l \mathbf{w}_l^{\mathcal{V}\mathcal{V}} + \mathbf{C}_l^\top \mathbf{w}_l^{\mathcal{E}\mathcal{V}} + \mathbf{b}_l^{\mathcal{V}}, \\ \mathbf{e}_{l+1} &= \mathbf{C}_l \mathbf{w}_l^{\mathcal{V}\mathcal{E}} + \mathbf{C}_l^\top \mathbf{w}_l^{\mathcal{E}\mathcal{E}} + \mathbf{b}_l^{\mathcal{E}}, \end{aligned} \tag{7}$$

where $\mathbf{w}_l^{\cdot\cdot} \in \mathbb{R}^d$ and $\mathbf{b}_l^{\cdot} \in \mathbb{R}^d$ are learnable parameters and bias vectors, respectively. Such a cross-learning process can be denoted by

$$[\mathbf{v}_{l+1}, \mathbf{e}_{l+1}] = C(\mathbf{v}_l, \mathbf{e}_l). \tag{8}$$

We also denote the final updated item embedding matrix as $\bar{V}$. Equipped with cross item-preference learning, the recommendation module can adaptively encode the item/entity knowledge transferred from the preference graph so that the correlation between two tasks can be learned. Note that we consider only lower-level layers in learning cross features. The reason is that lower-layer features can be generally shared by different tasks, and thus are better to be transferred [21, 43]. Higher-layer features tend to be specific to the targeted tasks. The final embeddings of items and entities after $\mathcal{L}$ layers can be represented by

$$\begin{aligned} \tilde{\mathbf{v}} &= C_v^{\mathcal{L}}(\mathbf{v}, \mathbf{e}), \\ \tilde{\mathbf{e}} &= C_e^{\mathcal{L}}(\mathbf{v}, \mathbf{e}), \end{aligned} \tag{9}$$

respectively, where $\mathcal{L}$ is the number of layers in cross-learning and set as $\mathcal{L} = 2$ by default.

4.3 Fairness-aware Preference Graph Embedding Learning

Preference graph embedding (**PGE**) aims at encoding both the properties of items and the preferences of users into entity embeddings so that the recommender can be aware of items' and users' knowledge through the abovementioned cross item-preference learning. We also implement the idea of fairness into PGE by considering how items are interacted by users with different attributes when sampling the triplets of the head, relation, and tail in PGE. Below we first present how to construct the preference graph, followed by the learning of entity embeddings with fairness-aware triplet sampling. We will utilize Figure 3 to describe the proposed FPGE learning in a step-by-step manner.

4.3.1 Preference Graph Construction. Users possessing some attributes may have a higher potential to interact with items with certain properties. We construct a *preference graph* to represent the relationships that user traits interact with item properties. The preference graph is utilized to encode the knowledge on the correlation between user attributes and item properties in the entity embeddings so as to benefit recommendation and provide fairness.

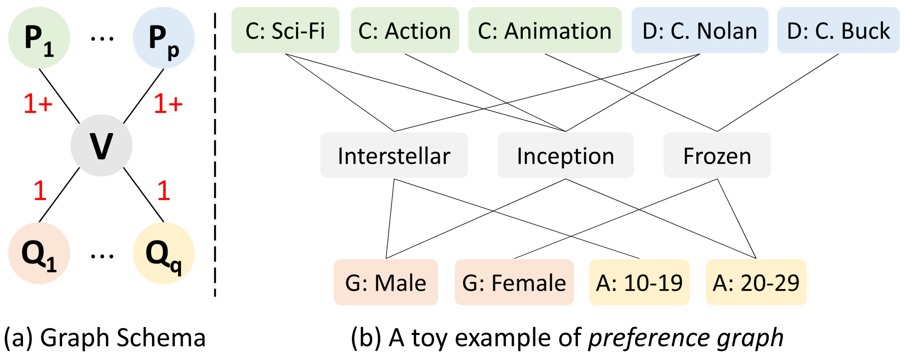


Fig. 4. Illustration of preference graphs. (a) The graph schema contains p item properties (P) and q user attributes (Q). In addition, "1+" indicates that one item entity can connect to *one or multiple* entities of an item property, and "1" means that one item entity can connect to *only one* entity of a user attribute. (b) In the toy example on the movie realm, "C", "D", "G", and "A" are categories, director, gender, and age, respectively. The first two are item properties, and the last two are user attributes.

At the first step of FPGE, as shown in Figure 3, we create the preference graph as a tri-partite graph structure, denoted by $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations, respectively. There are three types of entities in $\mathcal{E}$, including items, user attributes, and item properties. User attributes are those we described in Section 3. Item properties are metadata labels associated with items, such as categories, producers, and origin. There are two kinds of relations in $\mathcal{R}$. One is connecting items to user attributes, and the other is connecting items to item properties. In other words, there are no edges between items, between user attributes, and between item properties. An illustration of the preference graph is shown in Figure 4. The graph schema depicts p item properties (P) and q user attributes (Q). Every item entity can be connected with *only one* entity of a particular user attribute, and *one or multiple* entities of a particular item property.

In the toy example of Figure 4(b) on the movie realm, which contains 2 user attributes and 2 item properties. Each item entity connects to only one gender (G) value and only one age (A) value. Besides, each item entity can connect to one or multiple movie categories and directors. For instance, the movie "Inception" is connected to categories "Sci-Fi" and "Action."

The aim of connecting an item to a user attribute is to not only learn how their correlation, but also enables the incorporation of fairness in PGE learning, which will be described in the following part. However, items are not directly related to user attributes. For each item $v \in \mathcal{V}$ and each attribute $a \in A$, we connect v to only one of a's values $z^a_\star \in Z^a$ if $z^a_\star$ takes the *major* proportion among users who had ever interacted with v. The selection of major attribute value $z^a_\star$ for item v can be depicted by:

$$z^a_\star = \arg\max_{z^a \in Z^a} p_v(z^a), \tag{10}$$

where $p_v(z^a)$ is the adoption proportion introduced in Section 3.

4.3.2 *PGE Learning.* We consider PGE learning as a kind of knowledge graph embedding [35]. In PGE learning, entities in the preference graph can be divided into three types, *head* (h), *relation* (r), and *tail* (t), which can be treated as a triplet (h, r, t). While each item has a corresponding item entity in the preference graph, multiple layers of cross item-preference learning (Section 4.2) will first convert the original item embeddings to their latent representations of head entities $\mathbf{h}$, i.e., $\mathbf{h} = \tilde{\mathbf{e}} = C_e^{\mathcal{L}}(\mathbf{v}, \mathbf{e})$. In the meanwhile, the each relation r is also processed through $\mathcal{L}$ fully-connected neural network layers, given by $\mathbf{r} = \mathcal{M}^{\mathcal{L}}(\mathbf{r}_0)$, where $\mathbf{r}_0$ is the initialized one-hot encoding vector of relation IDs or types, and $\mathcal{M}^{\mathcal{L}}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$ is an $\mathcal{L}$-layer fully-connected neural network with trainable weights $\mathbf{W}$, bias $\mathbf{b}$, and the non-linear activation function $\sigma(\cdot)$. The process of learning PGE consists of three parts, *information passing*, *relational attention*, and *tail prediction.*

Information Passing. An entity can participate in one or multiple triplets, and thus can be regarded as an intermediate to pass information from one triplet to another in the preference graph. For example, in Figure 4(b), entity "Frozen" is involved in two triplets on item properties "C: Animation" and "D: C. Buck" and two triplets on user attributes "G: Female" and "A: 20–29." The last two can be indirectly depicted by the first two through information propagation from the first two to "Frozen", which is further passed to the last two. In other words, we encode user attributes and item properties (i.e., neighbors) into item entities in direct and indirect manners.

Let $\mathcal{N}_h = \{(h,r,t)|(h,r,t) \in \mathcal{G}\}$ be the set of triplets whose head is h, which is the item entity. As shown in Figure 3, at the third step of FPGE, we use the neighbors of entity h to represent h itself via linear combination

$$\mathbf{h} = \sum_{(h,r,t)\in\mathcal{N}_h} \alpha(h,r,t)\mathbf{t}, \tag{11}$$

where $\alpha(h,r,t)$ is the *relational attention weight* that determines how the edge (h,r,t) contributes to the representation of head h from tail t conditioned on relation r. We can also consider $\alpha(h,r,t)$ as a gate that decides how much information is being passed from t to h through r.

Relational Attention. We learn the relational attention weights by graph attention mechanisms [32, 36] at the second step of FPGE, as shown in Figure 3. The idea is to estimate the distance between entity h and tail t by projecting them into the space of relation r, and allow those head-tail pairs with shorter distance to have higher attention weights. Higher Similarity between vectors $\mathbf{h}$ and $\mathbf{t}$ conditioned on relation r leads to much information propagated from t to h. The derivation of relational attention is given by

$$\alpha(h,r,t) = (\mathbf{W}_r\mathbf{t})^\top \rho\,(\mathbf{W}_r\mathbf{h} + \mathbf{r}), \tag{12}$$

where ρ is the non-linear activation function *tanh*, and $\mathbf{W}_r$ is a matrix of learnable weights. Then we further normalize the attention weights over all triplets that entity h participates in based on the softmax function, given by

$$\alpha(h,r,t) = \frac{\exp(\alpha(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h} \exp(\alpha(h,r',t'))}. \tag{13}$$

The resultant attention scores are used in Equation (11) to highlight which item properties and user attributes provide stronger correlation signals to item entities.

Tail Prediction. At the last step of FPGE, as shown in Figure 3, we concatenate the derived head and relation embedding vectors $\mathbf{h}$ and $\mathbf{r}$ together. By feeding it into a $\mathcal{K}$-layer fully-connected neural network ($\mathcal{K} = 1$ by default), our target is to predict the tail embedding $\mathbf{t}$, given by $\hat{\mathbf{t}} = \mathcal{M}^{\mathcal{K}}(\mathbf{h},\mathbf{r})$, where $\hat{\mathbf{t}}$ is the predicted tail embedding. Since we will eventually put the PGE learning into the final loss function, for each triplet (h,r,t), the normalized inner product is used to generate a score b that measures the goodness of the PGE task, given by

$$b(h,r,t) = \sigma(\mathbf{t}^\top\hat{\mathbf{t}}), \tag{14}$$

where $\mathbf{t}$ is the real-value embedding vector of tail t.

4.3.3 Fairness-aware Triplet Sampling. We argue the original random triplet sampling brings *unfair* attribute information in entity embeddings. Past studies pointed out that the bias or unfairness of a recommender comes from how users with particular demographic attributes prefer to interact with some items [2, 9, 10], leading to imbalanced distributions of user attributes on items. Hence, random triplet sampling is biased to produce triplets containing entities of popular attributes, which further prohibit recommenders from being fair.

To remedy the bias and impose the fairness, we aim at making the item head embeddings be aware of fairness on user attributes through a proposed fairness-based triplet sampling (at the fourth step of FPGE, as shown in Figure 3), which consists of two phases. First, we adjust the probability of each triplet being sampled. Second, we devise an item head-based sampling strategy. Specifically, given the targeted set of attributes $\tilde{A}$ considered for the fairness, for every head h_v of item v, we first identify its set of neighboring entities $\mathcal{N}_{h_v}^{\tilde{A}}$, which belong to user attribute $a \in \tilde{A}$. For each tail entity $t_{z^a} \in \mathcal{N}_{h_v}^{\tilde{A}}$, the probability that triplet (h_v, r, t_{z^a}) being sampled is proportional to the reciprocal of the number of users with attribute value z^a who had interacted with item v via

$$\pi(h_v, r, t_{z^a}) = \frac{1/|\{u|\mathcal{A}^a(u) = z^a, u \in \mathcal{U}(v)\}|}{\sum_{z^{a'} \in Z^a} 1/|\{u|\mathcal{A}^a(u) = z^{a'}, u \in \mathcal{U}(v)\}|}. \quad (15)$$

Take the head entity of item "Interstellar" in Figure 4(b) as an example, assume 20 "Male" users 30 "10–19" users had interacted with "Interstellar", then we have $\pi(\text{“Interstellar”}, r, \text{“Male”}) = \frac{1/20}{1/20+1/30}$ and $\pi(\text{“Interstellar”}, r, \text{“10–19”}) = \frac{1/30}{1/20+1/30}$.

Based on Equation (15), if item v is frequently interacted by users with attribute value z^a, we lower down the probability of the corresponding triplet (h_v, r, t_{z^a}) being sampled. By doing so, we can have equal sampling possibility values for entities of all possible values of every attribute. After deriving the probabilities of all triplet (h_v, r, t_{z^a}), we normalize them to be in $[0, 1]$ for *positive* triplet sampling. The negative triplets are also sampled based on item heads. We randomly select an equal number of non-connected user-attribute tails to be the negative triplets.

4.4 Modeling Item-Item Correlation

The next items being recommended is influenced by the correlation between single items in the current item subsequence [46], in addition to their sequential effect. An existing study also extracted rules exhibiting a strong correlation between next and current items [23]. To exploit such item-item correlation, we model the correlation between the updated embeddings (i.e., after cross item-preference learning) of all items $\bar{\mathbf{V}}$ and the original item embeddings in the subsequence $\mathbf{S}_{u,l}$. The inner product is applied to obtain a correlation score, given by

$$\sum_{\mathbf{v}_j \in \mathbf{S}_{u,l}} \mathbf{v}_j^\top \cdot \bar{\mathbf{V}}, \quad (16)$$

where $\bar{\mathbf{V}} \in \mathbb{R}^{d \times N}$ is the updated item embedding matrix (after cross-learning in Section 4.2). This score reflects the correlation between each item in $\mathcal{S}_{1:t}^u$ and each candidate item $v \in \mathcal{V}$.

4.5 Prediction Layer

The prediction of the next items consists of three parts. First, we adopt the conventional matrix factorization [13] to capture the long-term interests of users. Second, we consider the interaction between users and the sequential features learned in Section 4.1 to model short-term interests of users. Third, the item-item correlation is incorporated in the prediction. For a given l th item subsequence, the predicted score of user u on item v can be represented by:

$$\hat{y}_{u,v} = \bar{\mathbf{u}}^\top \cdot \bar{\mathbf{v}} + \mathbf{s}_{u,l}^\top \cdot \bar{\mathbf{v}} + \sum_{\mathbf{v}_j \in \mathbf{S}_{u,l}} \mathbf{v}_j^\top \cdot \bar{\mathbf{v}}, \quad (17)$$

Table 1. Statistics of Datasets

	# Users	# Items	# Interactions	# PG triplets	# Seqs
Instagram	1,007	4,687	219,690	23,784	71,216
MovieLens-1M	619	2,347	125,112	14,368	83,408
Book-Crossing	384	14,910	70,696	17,232	20,232

Seqs denotes "sequences".

where $\bar{\mathbf{u}} = \mathcal{M}^{\mathcal{L}}(\mathbf{u}) \in \mathbb{R}^d$ is the updated user embedding after fully-connected layers, $\mathbf{u}$ is the input user embedding, $\bar{\mathbf{v}} \in \mathbb{R}^d$ is a column vector of the updated item embedding matrix $\bar{\mathbf{V}}$. We expect the true item v adopted by user u can lead to a higher prediction score $\hat{y}_{u,v}$.

4.6 Model Training

The overall loss function consists of two parts. One is the loss for user-item prediction in SR, and the other is the loss for FPGE. We optimize the SR part by **Bayesian Personalized Ranking** (**BPR**) objective [29], i.e., the pairwise ranking between positive and non-interacted items. In addition, we optimize the FPGE part by increasing the score for all positive triplets while decreasing the score for all negative triplets. The loss function is as follows:

$$\begin{aligned}\mathcal{J} =& \mathcal{J}_{SR} + \mathcal{J}_{FPGE} + \lambda \left\|\Theta\right\|_2^2 \\ =& \sum_{(u,v,\mathcal{S}_l^u,v') \in \mathcal{D}} -\log \sigma(\hat{y}_{u,v} - \hat{y}_{u,v'}) \\ & - \lambda_1 \left(\sum_{(h,r,t) \in \mathcal{G}} b(h,r,t) - \sum_{(h,r',t') \notin \mathcal{G}} b(h,r',t') \right) \\ & + \lambda_2 \left\|\Theta\right\|_2^2,\end{aligned} \tag{18}$$

where $\mathcal{S}_{1:t}^u$ denotes one t-length item subsequence of user u, v' is one non-interacted item, $\mathcal{D}$ is the entire dataset, (h, r', t') indicates negative triple sampling for efficient training, Θ contains all learnable parameters in the neural network, and λ_1 and λ_2 are the balancing hyperparameters. We utilize Adam [19] to adaptively adjust the learning rate during learning. In each training iteration, since our ultimate goal is for SR, we first repeat train the SR part for ϵ times ($\epsilon = 3$ by default), and train the FPGE part once.

5 EXPERIMENTS

We conduct a series of extensive experiments to answer the following four evaluation questions.

- **EQ1:** Can the proposed FairSR outperform state-of-the-art models in the sequential recommendation?
- **EQ2:** Is FairSR able to improve the fairness of recommended items, comparing to other fairness-aware models?
- **EQ3:** How does each component of FairSR contribute to the recommendation performance?
- **EQ4:** Is FairSR robust to the sensitivity of various hyperparameters?

5.1 Evaluation Setup

Datasets. Three datasets are employed in the evaluation. (a) **Instagram**: a user check-in records on locations [47] at three major urban areas, New York, Los Angeles, and London crawled via

Instagram API in 2015. Check-in locations are treated as items. (b) **MovieLens-1M**:[2] a widely-used benchmark dataset for movie recommendation. (c) **Book-Crossing**:[3] it contains explicit ratings (from 0 to 10) of books in the Book-Crossing community. Since MovieLens-1M and Book-Crossing are explicit feedback data, we follow MKR [34] to convert them into implicit feedback (i.e., 1 indicates that the user has rated the item and otherwise 0. The threshold of positive ratings is 4 for MovieLens-1M 9 for Book-Crossing. Since the main task is SR and we need user attributes for fairness, we preprocess the datasets by removing users without any attributes and users containing fewer than 4 interactions with items. The data statistics are summarized in Table 1. We have demographic attributes gender and age in both Instagram and Book-Crossing, and gender in MovieLens-1M. We take 10 years as the range of an attribute value for age. The protected attribute groups are composed of all combinations of attribute values in respective datasets.

Competing Methods. We compare several FiarSR with state-of-the-art methods and baselines. Their settings of hyperparameters are tuned by grid search on the validation set.

— **Caser**[4] [30] is a sequence embedding model that learns sequential features of items through convolution mechanisms.
— **RippleNet**[5] [33] is a memory-network-like approach that propagates user preferences on items via a knowledge graph.
— **SASRec**[6] [18] is a self-attention-based sequential model that utilizes the attention mechanism to identify relevant items and their correlation in entire item sequences.
— **MKR**[7] [34] (state-of-the-art) is a multi-task learning-based model that devises a task interaction learning to combine tasks of user-item matching and knowledge graph embedding.
— **HGN**[8] [22] (state-of-the-art) is a hierarchical gating network that learns the item subsequence embeddings through feature gating in long and short aspects, and models the item-item relations.
— **HGN+MKR** (state-of-the-art+state-of-the-art): we replace the recommendation module of MKR [34] with HGN [22] so that the knowledge graph embedding can be incorporated for a sequential recommendation. HGN+MKR is a very strong competitor as being capable of the power of both HGN and MKR.
— **FATR**[9] [49] is a fairness-aware tensor-based model that imposes *statistical parity* into item recommendation, i.e., ensuring similar probability distributions of item adoptions for users in different protected attribute groups.
— **FairSR-R** replaces fairness-aware triplet sampling with random sampling in FairSR. It serves as a control method to validate whether FairSR leads to fairness.
— **FairSR** is the full version of our proposed model.

Evaluation Metrics. For SR performance, we adopt in terms of *Precision@k* ($P@k$), *Recall@k* ($R@k$), and *NDCG@k* ($N@k$). To examine whether the recommended items exhibit the proposed *interaction fairness* defined in Section 3, we design the *difference of interaction fairness* ($DIF@k$) between the recommended items and the ground truth to be the metric, i.e., $DIF = \hat{IF} - IF$, where $\hat{IF}$ is the IF score generated from recommended items, and IF is the IF score of the corresponding

[2]https://grouplens.org/datasets/movielens/1m/.
[3]http://www2.informatik.uni-freiburg.de/~cziegler/BX/.
[4]https://github.com/graytowne/caser_pytorch.
[5]https://github.com/hwwang55/RippleNet.
[6]https://github.com/kang205/SASRec.
[7]https://github.com/hwwang55/MKR.
[8]https://github.com/allenjack/HGN.
[9]https://github.com/Zziwei/Fairness-Aware_Tensor-Based_Recommendation.

Table 2. Main Experimental Results of Precision, Recall, and NDCG for Sequential Recommendation in Three Datasets

	Instagram			MovieLens-1M			Book-Crossing		
	P@10	R@10	N@10	P@10	R@10	N@10	P@10	R@10	N@10
Caser	0.0166	0.0367	0.0199	0.0932	0.0832	0.0953	0.0102	0.0235	0.0125
SASRec	0.0264	0.0414	0.0291	0.0942	0.0862	0.0991	0.0193	0.0491	0.0211
RippleNet	0.0325	0.0401	0.0389	0.1305	0.1134	0.1387	0.0191	0.0843	0.0274
MKR	0.0287	0.0404	0.0281	0.1010	0.1194	0.1165	0.0287	0.1072	0.0294
HGN	0.0314	0.0457	0.0321	0.1146	0.1172	0.1317	0.0215	0.0235	0.0319
HGN+MKR	0.0387	0.0402	0.0450	0.1347	0.1292	0.1419	0.0311	0.1131	0.0345
FATR	0.0298	0.0326	0.0363	0.0915	0.1045	0.0653	0.0205	0.0954	0.0326
FairSR-R	**0.0473**	0.0414	**0.0492**	**0.1462**	**0.1395**	0.1414	**0.0462**	0.1383	**0.0435**
FairSR	0.0464	**0.0465**	0.0485	0.1389	0.1271	**0.1449**	0.0408	**0.1452**	0.0416

Bold and underline indicate the best and the second-best in each metric (column), respectively.

ground truth. Higher positive values of **Difference of Interaction Fairness** (**DIF**) indicate better fairness performance. The negative DIF value of a model means the fairness cannot get improved by that model.

Experimental Settings. The ratio of training, validation, and test sets is $6:2:2$. We repeat every experiment 10 times, and report the average results. We follow existing studies [22, 30] to fix the subsequence length $t = 5$ and $L = 8$, i.e., future length $g = L - t = 3$, by default, and will report the results of varying t and g. In FairSR, we apply a grid search for selecting proper hyperparameters using the validation set. Eventually we set $\lambda_1 = 1$, $\lambda_2 = 10^{-6}$, and $\epsilon = 3$ by default for all datasets. We examine how different hyperparameters affect the performance in the following. All experiments are conducted with PyTorch running on GPU machines (Nvidia GeForce GTX 1080 Ti).

5.2 Experimental Results

SR Performance Comparison. To answer **EQ1**, we present the results on SR performance shown in Table 2. We have the following findings. First, FairSR and FairSR-R consistently outperform the state-of-the-art methods and baselines in terms of precision, recall, and NDCG. The superiority indicates that although FairSR is originally devised to bring fairness into SR, it can still maintain and even improve the SR performance. Second, FairSR-R is slightly better than FairSR. Such a result is expectable because FairSR sacrifices popular (i.e., biased) attributes and items to bring fairness in recommended items. Third, among state-of-the-art methods, HGN+MKR is the most competitive, but is still worse than FairSR. This result implies that although HGN+MKR can incorporate knowledge graphs among items into SR, the modeling of convolution mechanisms, relational attention, personalized feature gating, and fairness-aware sampling in FairSR can further improve the performance. Fourth, among the metrics, FairSR-R generates more significant improvement on Precision, 22.2%, 8.5%, and 48.6% improvement over the most competitive state-of-the-art HGN+MKR. Such results prove the top-k items recommended by FairSR-R can accurately capture user preferences.

Fairness Comparison. To answer **EQ2**, we report the results on interaction fairness based on different models, as shown in Table 3. We can find that FairSR consistently produces the highest DIF scores over FairSR-R, the fairness baseline FATR, and other SR competing methods, and the superiority is obviously significant. In terms of DIF scores, the most competitive method is FATR. Although FATR leads to the second-best in Instagram and Book-Crossing datasets, our FairSR can still generate the highest DIF scores in all datasets, and FATR apparently has unsatisfying

Table 3. Main Experimental Results of the DIF in Three Datasets

	DIF@10		
	Instagram	MovieLens-1M	Book-Crossing
Caser	0.2293	0.0103	0.1031
SASRec	0.1835	<u>0.0158</u>	0.1246
RippleNet	0.1331	−0.0039	0.0526
MKR	0.1390	−0.0759	0.0684
HGN	0.3074	0.0098	0.1138
HGN+MKR	0.4583	−0.0823	0.0710
FATR	<u>0.6632</u>	0.0042	<u>0.1943</u>
FairSR-R	0.6015	−0.0964	0.1572
FairSR	**0.6979**	**0.0395**	**0.2217**

Bold and <u>underline</u> indicate the best and the second-best in each metric (column), respectively.

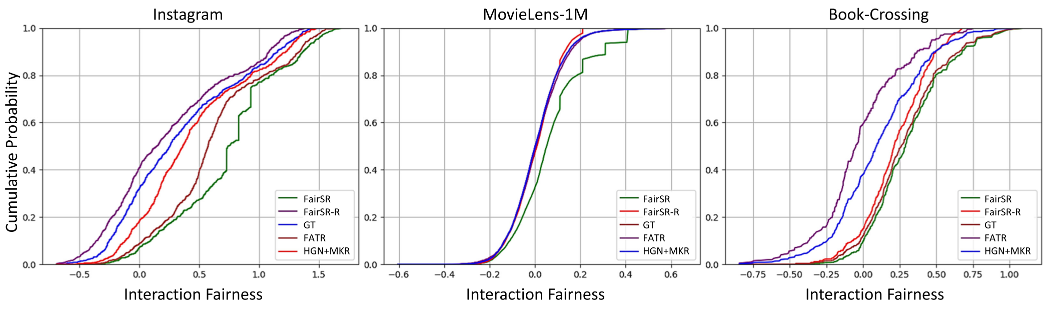


Fig. 5. Results on distributions of cumulative probability for IF, i.e., accumulated over users IF values, in Instagram data.

SR performance. While DIF is an aggregated statistic over users, we further plot the cumulative probability (y-axis) distributions on IF values (x-axis), i.e., accumulated by users' IF values from low to high. We compare IF cumulative probability distributions of FairSR, FairSR-R, the **ground truth** (**GT**), and HGN+MKR, and the results are shown in Figure 5. All of these results indicate that FairSR is able to not only improve SR performance, but also largely mitigate the unfairness coming from biased interactions between items and user attributes. In addition, the results also imply the proposed fairness-aware preference graph embedding truly takes effect. Moreover, it also provides strong evidence and positive feasibility on having satisfying and fair SR. On the other hand, the DIF values of competing methods are relatively low. We think the reason is existing SR tends to learn user preferences on popular items, which leads to biased user-item interactions. Although the fairness-aware RS method FATR can to some extent improve the DIF values, it still falls behind our proposed FairSR.

Ablation Analysis. To answer **EQ3**, we examine the contribution of each component in FairSR. In this experiment, we compare the FairSR full model with those replacing **fairness-aware sampling (-FS)** with random sampling (i.e., FairSR-R), **removing rational attention** (**-RA**), **removing convolution mechanisms** (**-Conv**), **removing personalized feature gating** (**-PFG**), and **removing FPGE** (**-FPGE**). We also remove both personalized feature gating and FPGE (**-PFG&FPGE**) from the full model since they are two main designs in this work. The results in Table 4 show that each component truly contributes to the full model. FPGE contributes most,

Table 4. Results on Recall and NDCG for Ablation Analysis

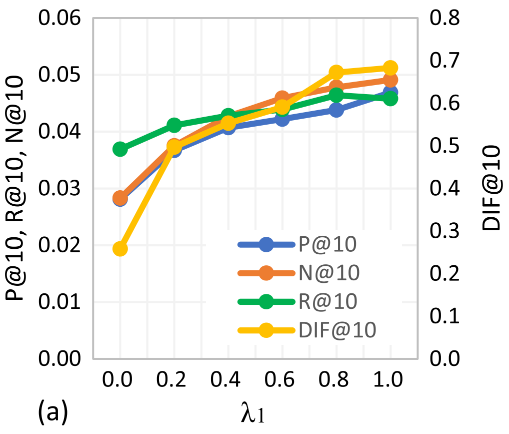
	Instagram		MovieLens-1M		Book-Crossing	
	R@10	N@10	R@10	N@10	R@10	N@10
FairSR	**0.0465**	0.0485	0.1271	**0.1449**	**0.1249**	0.0416
-FS	0.0414	0.0492	0.1156	0.1414	0.1114	**0.0435**
-RA	0.0394	0.0469	**0.1295**	0.1379	0.1172	0.0381
-Conv	0.0439	0.0479	0.1093	0.1389	0.0962	0.0401
-PFG	0.0414	**0.0516**	0.1194	0.1487	0.0917	0.0372
-FPGE	0.0375	0.0294	0.1039	0.1075	0.0472	0.0199
-PFG&FPGE	0.0332	0.0340	0.0914	0.1064	0.0351	0.0128

Table 5. The Effect of the Sequence Length t and g for FairSR

	Instagram		MovieLens-1M		Book-Crossing	
	R@10	N@10	R@10	N@10	R@10	N@10
$t = 3$ & $g = 1$	0.0375	0.0342	0.1194	0.1328	0.1134	0.0383
$t = 3$ & $g = 2$	0.0386	0.0353	0.1202	0.1377	0.1159	0.0394
$t = 3$ & $g = 3$	0.0409	0.0373	0.1214	0.1401	0.1217	0.0417
$t = 5$ & $g = 1$	0.0403	0.0385	0.1235	0.1392	0.1207	0.0395
$t = 5$ & $g = 2$	0.0451	0.0436	0.1229	0.1413	0.1221	0.0403
$t = 5$ & $g = 3$	**0.0489**	**0.0473**	0.1247	**0.1459**	**0.1253**	**0.0424**
$t = 8$ & $g = 1$	0.0433	0.0423	0.1301	0.1411	0.1229	0.0403
$t = 8$ & $g = 2$	0.0468	0.0439	0.1319	0.1427	0.1217	0.0413
$t = 8$ & $g = 3$	0.0479	0.0467	**0.1337**	0.1449	0.1244	0.0417

i.e., leading to the largest performance loss, indicating that FairSR highly relies on FPGE to encode how users with different attributes interact with items. In addition, the model is significantly damaged if both PFG and FPGE are removed (**-PFG&FPGE**). Such a result again verifies the usefulness of our main technical designs. By looking into the details on the removal of each component in Table 4, we can have the following two deeper insights. First, removing the **-FS** from the full FairSR model can improve the performance of sequential recommendation in the cases of NDCG@10 on Instagram and NDCG@10 on Book-Crossing. We think such a performance drop off "**-FS**" is reasonable and acceptable. FairSR aims at striking a balance between SR performance and interaction fairness. The model with fairness-aware sampling makes the training totally focus on generating good SR performance, which is reflected on the results of "**-FS**". Second, we can find that in the case of Recall@10 on MovieLens, the SR performance gets slightly improved the FairSR model without **-RA**. The potential reason can be that there is only one attribute "gender" considered in the construction of the preference graph. Hence, the relational attention mechanism cannot work well to differentiate items associated with various properties in the preference graph.

Hyperparameter Sensitivity. To answer **EQ4**, we present the effect of three hyperparameters: the contribution of FPGE loss λ_1 in Equation (18), the frequency of SR training ϵ in each iteration, and the lengths of training and testing sequences t and g. The results are reported in Figure 6 and Table 5. We can first find that higher λ_1 values lead to better SR performance and higher interaction fairness. Such a result implies the proposed fairness-aware preference graph embedding is able to bring a positive effect on both SR and fairness. As for ϵ, paying too much attention to the SR task (i.e., larger ϵ) weakens the contribution from the FPGE task that simultaneously models interactions between items and attributes and alleviates unfairness via triplet sampling. A proper ϵ



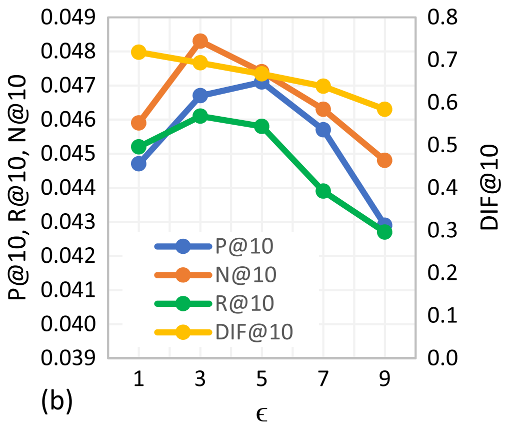


Fig. 6. The studies of hyperparameter sensitivity of FairSR for (a) the weight of FPGE loss λ_1 and (b) the SR training frequency ϵ.

is suggested to be 3 that strikes a balance between SR and FPGE in SR performance and fairness. On the other hand, we can have the following observations. The length with $t = 5$ & $g = 3$ produces better performance. Both increasing t with fixed g and increasing g with fixed t lead to performance improvement. This indicates that we need more training items (higher t) to learn short-term user interest, and the given item sequence can determine multiple preferred items (higher g).

6 CONCLUSION

While the filter bubble effect widely happens to online services, it is crucial to have it considered in recommender systems. To alleviate the filter bubble effect, this article proposes and solves a novel fairness-aware sequential recommendation task. We define a new metric, interaction fairness, which reflects the degree that items are equally interacted by users with different protected attribute groups. A multi-task learning approach, FairSR, is developed to not only learn personalized sequential features, but also model fairness through embedding items and attributes into the same space via a fair triplet sampling. Experimental results on three datasets prove the effectiveness of FairSR over state-of-the-art SR models and other fairness-aware RS, and exhibit promising interaction fairness.

There are several future extensions based on FairSR. First, the current preference graph presumes that there is no correlation between item properties and between user attributes. We are seeking to automatically learn their underlying relationships. Second, scalability is an essential issue for recommender systems. The next step is to have FairSR scalable to millions of interactions between users and items. Third, in a realistic setting of recommender systems, new users and new items are continuously coming. Hence, we plan to have FairSR to be capable of inductive learning.

REFERENCES

[1] Himan Abdollahpouri. 2019. Popularity bias in ranking and recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society.* 529–530.

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The Impact of Popularity Bias on Fairness and Calibration in Recommendation. arXiv:1910.05755. Retrieved from https://arxiv.org/abs/1910.05755.

[3] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2212–2220.

[4] Avishek Bose and William Hamilton. 2019. Compositional fairness constraints for graph embeddings. In *Proceedings of the 36th International Conference on Machine Learning*. 715–724.

[5] Renqin Cai, Hongning Wang, Jibang Wu, Chong Wang, and Aidan San. 2021. Category-aware collaborative sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[6] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of the World Wide Web Conference.* 151–161.

[7] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Jointly non-sampling learning for knowledge graph enhanced recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 189–198.

[8] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70.* 933–941.

[9] Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. 2018. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, Vol. 81. 172–186.

[10] Golnoosh Farnadi, Pigi Kouki, Spencer K. Thompson, Sriram Srinivasan, and Lise Getoor. 2018. A fairness-aware hybrid recommender system. In *Proceedings of 2nd FATREC Workshop: Responsible Recommendation.*

[11] Ming-Han Feng, Chin-Chi Hsu, Cheng-Te Li, Mi-Yen Yeh, and Shou-De Lin. 2019. MARINE: Multi-Relational network embeddings with relational proximity and node attributes. In *Proceedings of the World Wide Web Conference.* 470–479.

[12] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-Aware ranking in search and recommendation systems with application to LinkedIn Talent Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2221–2231.

[13] 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.

[14] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* 843–852.

[15] Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of International Conference on Learning Representations.*

[16] Cheng Hsu and Cheng-Te Li. 2021. RetaGNN: Relational temporal attentive graph neural networks for holistic sequential recommendation. In *Proceedings of the Web Conference.*

[17] Jyun-Yu Jiang, Cheng-Te Li, Yian Chen, and Wei Wang. 2018. Identifying users behind shared accounts in online streaming services. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 65–74.

[18] W. Kang and J. McAuley. 2018. Self-Attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining.* 197–206.

[19] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations.*

[20] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1269–1278.

[21] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning.* 97–105.

[22] 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.

[23] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *Proceedings of the World Wide Web Conference.* 1210–1221.

[24] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2022. A survey on bias and fairness in machine learning. *ACM Comput. Surv.* 54, 6, Article 115 (July 2022).

[25] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. 2014. Exploring the filter bubble: The effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide Web.* 677–686.

[26] Eli Pariser. 2011. *The Filter Bubble: What The Internet Is Hiding From You.* Penguin Books Limited.

[27] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards fair graph embedding. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence.* 3289–3295.

[28] Bashir Rastegarpanah, Krishna P. Gummadi, and Mark Crovella. 2019. Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining.* 231–239.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence.* 452–461.

[30] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining.* 565–573.

[31] Riku Togashi, Mayu Otani, and Shin'ichi Satoh. 2021. Alleviating cold-start problems in recommendation through pseudo-labelling over knowledge graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining.* 931–939.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations.*

[33] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* 417–426.

[34] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task feature learning for knowledge graph enhanced recommendation. In *Proceedings of the World Wide Web Conference.* 2000–2010.

[35] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[36] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 950–958.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 165–174.

[38] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference.*

[39] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence.* 5329–5336.

[40] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of The Web Conference 2020.* 99–109.

[41] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining.* 495–503.

[42] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 2925–2934.

[43] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Proceedings of the Advances in Neural Information Processing Systems.* 3320–3328.

[44] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-Order attentive ranking model for sequential recommendation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence.* 5709–5716.

[45] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining.* 582–590.

[46] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys* 52, 1 (2019), 1–38.

[47] Yang Zhang, Mathias Humbert, Tahleen Rahman, Cheng-Te Li, Jun Pang, and Michael Backes. 2018. Tagvisor: A privacy advisor for sharing hashtags. In *Proceedings of the 2018 World Wide Web Conference.* 287–296.

[48] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2021. Cold-start sequential recommendation via meta learner. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence.*

[49] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-Aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* 1153–1162.

Received May 2021; revised August 2021; accepted October 2021