Parallel Split-Join Networks for Shared Account Cross-Domain Sequential Recommendations

Wenchao Sun, Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten de Rijke

Abstract—Sequential recommendation is a task in which one models and uses sequential information about user behavior for recommendation purposes. We study sequential recommendation in a context in which multiple individual users share a single account (i.e., they have a shared account) and in which user behavior is available in multiple domains (i.e., recommendations are cross-domain). These two characteristics bring new challenges on top of those of the traditional sequential recommendation task. First, we need to identify the behavior associated with different users and different user roles under the same account in order to recommend the right item to the right user role at the right time. Second, we need to identify behavior in one domain that might be helpful to improve recommendations in other domains. We study *shared account cross-domain sequential recommendation* and propose a **p**arallel **s**plit-**j**oin **Net**work (Parallel Split-Join Network (PSJNet)), a parallel modeling network to address the two challenges above. We use "split" to address the challenge raised by shared accounts; PSJNet learns role-specific representations and uses a gating mechanism to filter out, from mixed user behavior, information of user roles that might be useful for another domain. In addition, "join" is used to address the challenge raised by the cross-domain setting; PSJNet learns cross-domain representations by combining the information from "split" and then transforms it to another domain. We present two variants of PSJNet: PSJNet-I and PSJNet-II. PSJNet-I is a "split-by-join" framework that splits the mixed representations to get role-specific representations and joins them to obtain cross-domain representations at each timestamp simultaneously. PSJNet-II is a "split-and-join" framework that first splits role-specific representations at each timestamp, and then the representations from all timestamps and all roles are joined to obtain cross-domain representations. We concatenate the in-domain and cross-domain representations to compute a recommendation score for each item. Both PSJNet-I and PSJNet-II can simultaneously generate recommendations for two domains where user behavior in two domains is synchronously shared at each timestamp. We use two datasets to assess the effectiveness of PSJNet. The first dataset is a simulated shared account cross-domain sequential recommendation dataset obtained by randomly merging the Amazon logs from different users in the movie and book domains. The second dataset is a real-world shared account cross-domain sequential recommendation dataset built from smart TV watching logs of a commercial organization. Our experimental results demonstrate that PSJNet outperforms state-of-the-art sequential recommendation baselines in terms of MRR and Recall.

Index Terms—Parallel modeling, shared account recommendation, cross-domain recommendation, sequential recommendation

1 Introduction

It is hard to apply traditional recommendation methods such as, e.g., collaborative filtering (CF)-based methods [2] or matrix factorization (MF)-based models [3], [4], in recommendation scenarios in which user profiles may be absent. This happens when users are not logged in or the recommender system does not track user-ids. For this reason, *sequential recommendation* (SR) has been proposed for session-based recommendations [5]. Compared with traditional recommendations, SR has natural advantages when it comes to sequential dynamics [6], i.e., SR methods may generate different lists of recommended items at different timestamps. The goal of SR is to generate recommendations based on a sequence of user behavior (e.g., a sequence of songs listened to, videos watched, or products clicked), where interactions are usually grouped by same time frame [7]. SRs have a wide range of applications in many domains such as e-commerce, job websites, music and video recommendations [8]. nd users usually have specific goals during the process, such as finding a good restaurant in a city, or listening to a song of a certain style or mood [9].

Early studies into SR are mostly based on Markov chains (MCs) [10] or Markov decision processes (MDPs) [8]. Sequences of items are considered as states and a state-transition matrix or function is learned to generate recommendations. In this way, the dynamic characteristics of SR are taken into account. However, because the states in a MC- or Markov decision process (MDP)-based method correspond to sequences of items, the state-transition matrix or function

- *Wenchao Sun, Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Zhaochun Ren, and Jun Ma are with the Information Retrieval Lab, School of Computer Science and Technology, Shandong University, Qingdao 266237, China. E-mail: swc18769786189@163.com, muyang0331@gmail.com, {renpengjie, chenzhumin, zhaochun.ren, majun}@sdu.edu.cn, yu.jie.lin@outlook.com.*
- *Maarten de Rijke is with the University of Amsterdam, 1012, WX, Amsterdam, The Netherlands. E-mail: m.derijke@uva.nl.*

Manuscript received 14 April 2021; revised 10 November 2021; accepted 21 November 2021. Date of publication 13 December 2021; date of current version 7 March 2023.
This work was supported in part by the Natural Science Foundation of China under Grants 62102234, 61972234, 61902219, and 62072279, the Natural Science Foundation of Shandong Province under Grants ZR2021QF129 and ZR202102230192, the Key Scientific and Technological Innovation Program of Shandong Province under Grant 2019JZZY010129, the National Key R&D Program of China under Grant 2020YFB1406704, the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, the Tencent WeChat Rhino-Bird Focused Research Program under Grant JR-WXG-2021411, and the Fundamental Research Funds of Shandong University.
(Corresponding author: Pengjie Ren.)
Recommended for acceptance by M. Wang.
Digital Object Identifier no. 10.1109/TKDE.2021.3130927

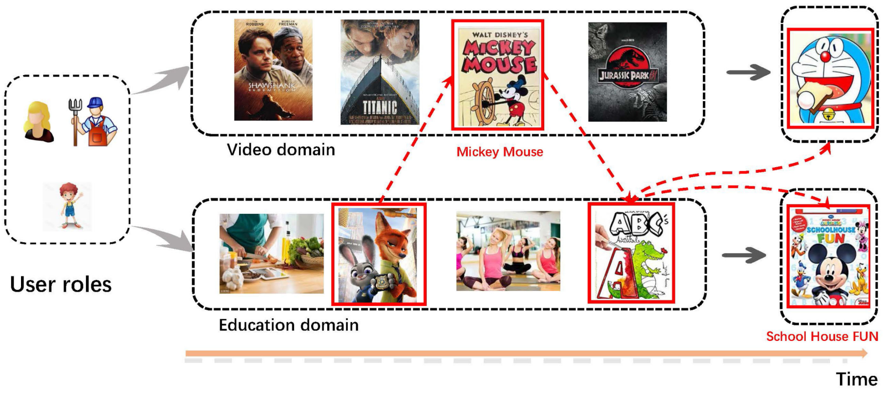

Fig. 1. The smart TV scenario provides a natural example of *shared account cross-domain sequential recommendation* (SAC-SR). Here, members of a family (the "user roles") share the same account. The *video* domain contains various movies, TV series, cartoons, etc. The *education* domain contains educational programs and technical tutorials, etc. *Boxed* items reflect similar user interests. Red lines show the interactions and connections between user behavior in the two domains.

quickly becomes unmanageable in realistic scenarios [11]. Recurrent neural networks have demonstrated their effectiveness in sequence modeling in the field of natural language processing. Motivated by this, recent studies have introduced recurrent neural networks (RNNs) into SR [5], [12], [13] and today RNN-based models have been widely adopted in the context of SR. Various RNN architectures have been proposed to enhance SR, e.g., to make SRs context-aware [14], personalized [9], or deal with repeat behavior [15]. However, so far RNN-based methods focus on a single domain and none simultaneously considers the shared account and cross-domain scenarios.

In this paper, we study SR in a particularly challenging context, *shared account cross-domain sequential recommendation* (SAC-SR). In this context multiple individual users share a single account (i.e., they have a *shared account*) and user behavior is recorded in multiple domains (i.e., recommendations should be *cross-domain*). In the shared account setting multiple users take on multiple 'user roles' under the same account, where user roles do not necessarily represent specific users. For example, in the smart TV recommendation scenario depicted in Fig. 1, members of a family correspond to different user roles, e.g., "parents", "children", and they share a single account to watch videos.

The existence of shared accounts makes it harder to generate relevant recommendations, because the behavior of multiple user roles is mixed together. Since, user roles are latent, they do not have to be consistent with the real number of people in a family. We consider user roles instead of users because we assume that relevance of a recommendation depends on user's role, not on their individual identity as a user.

We consider the cross-domain task because it is a common phenomenon in practice: users use different platforms to access domain-specific services in daily life. For example, many smart TV platforms use different channels to provide different services, e.g., a video channel (domain) that offers movies or TV series and an educational channel that offers educational material, as depicted in Fig. 1. User behavior in one domain may be helpful for improving recommendations in another domain [16], [17] because user behavior in different domains may reflect similar user interests. For example, as illustrated in Fig. 1, videos like "Mickey Mouse" in the video domain might help to predict the next item "School House Fun" in the educational domain because they reflect the same interest in the Disney cartoon character "Mickey Mouse" presumably by a child in this family. Although leveraging user behavior information from another domain may provide useful information to help improve the recommendation performance, this type of transfer is non-trivial because the behavior of multiple user roles is mixed and this may introduce noise. This raises an interesting challenge, namely how to identify behavior from one domain that might be helpful to improve recommendations in other domains while minimizing the impact of noisy information?

In prior work that focuses on shared accounts, a common approach is to capture user interests by extracting latent features from high-dimensional spaces that describe the relationships among user roles under the same account [18], [19]. And in prior work on the cross-domain task, one common solution is to aggregate information from two domains [20], [21], while another is to transfer knowledge from the source domain to the target domain [22]. These methods cannot be directly applied to SAC-SR: either important sequential characteristics of SR are largely ignored or they rely on explicit user ratings, which are usually unavailable in SR. We have introduced an architecture (π-Net) that addresses the above issues by simultaneously generating recommendations for two domains where user behavior from two domains is synchronously shared at each timestamp [1].

In this work, we generalize over π-Net with a more general framework, the **P**arallel **S**plit-**J**oin **Net**work (PSJNet), that introduces the "split" and "join" concepts to address the shared account and cross-domain characteristics in shared account cross-domain sequential recommendation (SAC-SR). "Split" is used to identify behavior of different user roles, where we employ a gating mechanism to extract, from mixed user behavior, role-specific representations containing information of user roles that might be useful for another domain. "Join" is used to discriminate and combine useful user behavior; we learn cross-domain representations by combining the information from "split" and then adopting it to another domain.

Specifically, PSJNet is organized in four main modules, a *sequence encoder*, a *split unit*, a *join unit*, and a *hybrid recommendation decoder*. The *sequence encoder* module encodes the current sequence of mixed user behavior from each domain into a sequence of in-domain representations. Then, depending on how "split" and "join" are implemented, we present two PSJNet variants, PSJNet-I and PSJNet-II. PSJNet-I, which corresponds to π-Net, employs a "Split-by-Join" scheme where it splits the mixed representations to get role-specific representations and joins them to get cross-domain representations at each timestamp simultaneously. We reformulate the shared account filter unit (SFU) and the cross-domain transfer unit (CFU) in π-Net as a split-by-join unit in PSJNet-I. The split-by-join unit does exactly the same thing as SFU and CFU. PSJNet-II employs a "Split-and-Join" scheme, where it first splits role-specific representations at each timestamp, and then the representations from all timestamps and all roles are joined to obtain cross-domain representations. For both variants, "split" and "join" are operated in a parallel recurrent way, which means that they can synchronously share information across both domains at each timestamp. Finally, the *hybrid recommendation decoder* module estimates the recommendation scores for each item based on the information from both domains, i.e., the in-domain representations from the target domain and the cross-domain representations from the complementary domain. During learning, PSJNet is jointly trained on two domains in an end-to-end fashion.

To assess the effectiveness of PSJNet, we need datasets that exhibit both shared account and cross-domain characteristics. To the best of our knowledge, there is no such real-world dataset that is publicly available. We construct two datasets for SAC-SR. The first dataset is a simulated SAC-SR dataset obtained by randomly merging the logs from different users in the movie and book domains from a well-known Amazon dataset.[1] Although the dataset satisfies our experimental requirements, merged user behavior is not realistic. Therefore, we build a second dataset from smart TV watching logs of a commercial company, which is a real-world SAC-SR scenario. We release both datasets to facilitate future research. We carry out extensive experiments on both datasets. The experimental results show that PSJNet outperforms state-of-the-art baselines in terms of MRR and Recall. We also conduct ablation studies to show that the proposed parallel "split" and "join" schemes are effective and useful for SAC-SR.

The additional contributions of this paper compared to our previous work [1] are:

- We present the PSJNet framework, which introduces the "split" and "join" concepts to address the shared account and cross-domain characteristics of SAC-SR.
- We reformulate the previous proposal π-Net as PSJNet-I within the PSJNet framework, and propose a variant PSJNet-II that further improves the performance.
- We carry out experiments on two datasets for SAC-SR. One is constructed by simulating shared account characteristics on a public dataset, the other is a real-world dataset. We conduct additional experiments on these two datasets to show the effectiveness of the two PSJNet variants.

2 Related Work

We consider three types of related work: sequential recommendations, shared account recommendations, and cross-domain recommendations.

1. http://jmcauley.ucsd.edu/data/amazon/

2.1 Sequential Recommendation

It is hard to capture sequential dynamics in recommendation scenarios with classical recommendation methods such as MF- or CF-based methods. Instead, dedicated methods have been developed for SR or next basket recommendation.

2.1.1 Traditional Methods

The traditional approaches for SR are mostly based on MCs [10] or Markov decision processes (MDPs) [8] to predict a user's next action given their last action [23]. Zimdars *et al.* [10] are the first to propose MCs for web page recommendation. They investigate how to extract sequential patterns to learn the next state using probabilistic decision-tree models. To further improve the performance, Shani *et al.* [8] propose an MDP-based recommendation method, where the next recommendation can be computed through the transition probabilities among items. To combine the advantages of MF and MC-based methods, Rendle *et al.* [24] propose a method based on personalized transition graphs over an underlying MC. The proposed method subsumes both a common MC and the normal MF model. Chen *et al.* [25] take playlists as an MC, and propose logistic Markov embeddings to learn representations of songs for playlist prediction. Lu *et al.* [26] argue that source domain data is not always consistent with the observations in the target domain, which may misguide the target domain recommendation. They propose a criterion based on empirical prediction error and its variance to better capture the consistency across domains in CF settings.

All of the MC or MDP-based sequential recommendation methods mentioned above show improvements by modeling sequential dynamics. A major limitation they share is that they can only consider a very short sequence (e.g., the most recent five items in [8]), because the state space quickly becomes unmanageable when taking long sequences into account [11].

2.1.2 Deep Learning-Based Methods

Recently, RNNs have been devised to model variable-length sequential data [9], [27], [28], [29], [30], [31], [32], [33]. Hidasi *et al.* [5] are the first to apply RNNs to sequential recommendation and achieve significant improvements over traditional methods. They utilize session-parallel mini-batch training and employ ranking-based loss functions to train the recommendation model. In later work, they propose data augmentation techniques to improve the performance of RNNs [34].

Contextual information has proved to be very important for behavior modeling in traditional recommendations [35]. Liu *et al.* [14] incorporate contextual information into SR and propose a context-aware RNN model. Instead of using the constant input matrix and transition matrix from conventional RNN models, their CA-RNN employs adaptive matrices. The authors use context-specific input matrices to capture external conditions under which user behavior happens, such as time, location, and weather. They also use context-specific transition matrices to capture how the length of time intervals between adjacent behavior in historical sequences affects the transition of global sequential features. Tang and Wang [36] propose a convolutional sequence embedding recommendation model for personalized top-n sequential recommendation to address the more recent items where they argue that more recent items in a sequence have a larger impact on the next item. Kang and Mcauley [37] propose a self-attention based sequential model to capture both the long-term semantics and relatively few actions. Li *et al.* [38] explicitly model the timestamps of interactions to explore the influence of time intervals on next item prediction. Luo *et al.* [39] predict the intent of the current session by investigating neighborhood sessions. Wang *et al.* [40] develop a next-item recommendation framework empowered by sequential hypergraphs.

More recently, Li *et al.* [41] propose a transformer-based structured intent-aware model that first extracts intents from sequential contexts, and then adopts an intent graph to capture the correlations among user intents. Wang *et al.* [42] consider user-item relationships at the finer granularity of user intents and generate disentangled representations. Zhang *et al.* [43] highlight the importance of recommender retraining research and formulate the sequential retraining process as an optimizable problem. Wu *et al.* [44] explore self-supervised learning on a user-item graph to improve the accuracy and robustness for recommendation. Wang *et al.* [45] explore intents behind a user-item interaction by using auxiliary item knowledge.

Memory enhanced RNNs have been well studied for SR recently [46]. Wang *et al.* [47] propose a RNN model with two parallel memory modules: one to model a user's own information in the current sequence and the other to exploit collaborative information in neighborhood sequences [48].

2.2 Shared Account Recommendation

Most recommender systems assume that an account in the data represents a single user. However, multiple users often share a single account. A typical example is a smart TV account for the whole family.

Previous approaches to shared account recommendations typically first identify users and then make personalized recommendations [18], [19]. Zhang *et al.* [49] are the first to study user identification, in which they use linear subspace clustering algorithms; they recommend the union of items that are most likely to be rated highly by each user. Bajaj and Shekhar [50] propose a similarity-based channel clustering method to group similar channels for IPTV accounts, and they use the Apriori algorithm to separate users that are merged under a single account. After that, they use personal profiles to recommend additional channels to the account. Wang *et al.* [51] assume that different users consume services in different periods. They decompose users based on mining different interests over different time periods from consumption logs. Finally, they use a standard User-KNN method to generate recommendations for each identified user. Jiang *et al.* [52] propose an unsupervised learning-based framework to identify users and differentiate the interests of users and group sessions by users. They construct a heterogeneous graph to represent items and use a normalized random walk to create item-based session embeddings. A hybrid recommender is then proposed that linearly combines the account-level and user-level item recommendation by employing Bayesian personalized ranking matrix factorization [53].

2.3 Cross-Domain Recommendation

Cross-domain recommendation concerns data from multiple domains, which has proven to be helpful for alleviating the cold start problem [54], [55] and data sparsity issues [56], [57]. There is an assumption that there exists overlap in information between users and/or items across different domains [58].

2.3.1 Traditional Methods

There are two main ways for dealing with cross-domain recommendations [59]. One is to aggregate knowledge between two domains. Tang *et al.* [60] propose a cross-domain topic learning model to address three challenges in cross-domain collaboration recommendation: sparse connections (cross-domain collaborations are rare); complementary expertise (cross-domain collaborators often have different expertise and interest) and topic skewness (cross-domain collaboration topics are focused on a subset of topics). Do *et al.* [61] propose to discover both explicit and implicit similarities from latent factors across domains based on matrix tri-factorization. Chen *et al.* [62] exploit the users and items shared between domains as a bridge to link different domains by embedding all users and items into a low-dimensional latent space between different domains. Liu *et al.* [63] utilize both MF and an attention mechanism for fine-grained modeling of user preferences; the overlapping cross-domain user features are combined through feature fusion.

The other approach to cross-domain recommendation is to transfer knowledge from the source domain to the target domain. Hu *et al.* [17] propose tensor-based factorization to share latent features between different domains. Doan and Sahebi [64] propose a transition-based cross-domain collaborative filtering method to capture both within- and between-domain transitions of user feedback sequences. Zhang *et al.* [65] propose a method that not only transfers an item's learned latent factors, but also selectively transfers user's latent factors.

2.3.2 Deep Learning-Based Methods

Deep learning is well suited to transfer learning as it can learn high-level abstractions among different domains [66], [67], [68]. Hu *et al.* [21] propose a model using a cross-stitch network [69] to learn complex user-item interaction relationships based on neural collaborative filtering [20]. Zhuang *et al.* [22] propose a novelty-seeking model to fully characterize users' novelty-seeking trait so as to obtain a better performance across domains. Wang *et al.* [16] are the first to introduce the problem of cross-domain social recommendations; they combine user-item interactions in information domains (such as online travel planning) and user-user connections in social network services (such as Facebook or Twitter) to recommend relevant items of information domains to target users of social domains; user and item attributes are leveraged to strengthen the embedding learning. Chen *et al.* [70] apply multi-level graph convolutions to a cross-platform account matching task. Xia *et al.* [71] model session-based data as a hypergraph and propose a hypergraph convolutional network to improve the performance of session-based recommendation. Guo *et al.* [72] also propose a graph-based solution to model multiple associations and structure-aware domain knowledge.

Although the methods proposed in the studies listed above have been proven to be effective in many applications, they either cannot be applied to sequential recommendations or do not consider the shared account or cross-domain characteristics. In our previous work, we have proposed π-Net in order to address shared account and cross-domain characteristics in sequential recommendations by extracting information of different user roles under the same account and transferring it to a complementary domain at each timestamp [1]. In this work, we present a more general framework called PSJNet: π-Net can be viewed as a particular instantiation of PSJNet and we propose another instantiation that further improves the recommendation performance over π-Net.

3 Method

In this section, we first provide a formulation of the SAC-SR problem. Then, we introduce PSJNet and describe two instantiations of the framework. For each variant, we first give a high-level introduction and describe each component in detail.

3.1 Shared Account Cross-Domain Sequential Recommendation

We represent a cross-domain behavior sequence (e.g., watching videos, reading books) from a shared account as $S = \langle A_1, B_1, B_2, \ldots, A_i, \ldots, B_j, \ldots \rangle$, where $A_i \in \mathbb{A}$ $(1 \le i \le N)$ is the index of a single consumed item in domain A; $\mathbb{A}$ is the set of all items in domain A; $B_j \in \mathbb{B}$ $(1 \le j \le M)$ is the index of a single consumed item in domain B; $\mathbb{B}$ is the set of all items in domain B; N and M are the number of items in the sequences from domain A and B, respectively. Given S, SAC-SR tries to predict the next item by computing the recommendation probabilities for all candidates in two domains respectively, as shown in Eq. (1):

$$\begin{aligned} P(A_{i+1} \mid S) &\sim f_A(S) \\ P(B_{j+1} \mid S) &\sim f_B(S), \end{aligned} \tag{1}$$

where $P(A_{i+1} \mid S)$ denotes the probability of recommending the item A_{i+1} that will be consumed next in domain A. Also, $f_A(S)$ is the model or function used to estimate $P(A_{i+1} \mid S)$. Similar definitions apply to $P(B_{j+1} \mid S)$ and $f_B(S)$.

The main differences between SAC-SR and traditional SR are two-fold. First, in SAC-SR, S is generated by multiple users (e.g., family members) while it is usually generated by a single user in SR. Second, SAC-SR considers information from both domains for the particular recommendations in one domain, i.e., S is a mixture of behavior from multiple domains. In this paper, we only consider two domains but the ideas easily generalize to multiple domains.

Next, we will describe two PSJNet variants in detail. The key idea of PSJNet is to learn a recommendation model that can first extract the information of some specific user roles from domain A, and then transfer the information to domain B, and combine it with the original information from domain B to improve the recommendation performance for domain B, and vice versa. This process is achieved in a parallel way, which means that the information from both domains is shared recurrently.

3.2 Sequence Encoder

Both variants of PSJNet that we consider use the same sequence encoder. Like previous studies [5], [9], [34], we use a RNN to encode a sequence S. Here, we employ two separate gated recurrent units (GRUs) as the recurrent units to encode the items from domain A and domain B respectively. And the GRU is given as follows:

$$\begin{aligned} z_t &= \sigma(W_z[emb(x_t), h_{t-1}]) \\ r_t &= \sigma(W_r[emb(x_t), h_{t-1}]) \\ \widetilde{h}_t &= \tanh(W_{\widetilde{h}}[emb(x_t), r_t \odot h_{t-1}]) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t, \end{aligned} \tag{2}$$

where W_z, W_r, and $W_{\widetilde{h}}$ are weight matrices; $emb(x_t)$ is the item embedding of item x at timestamp t; and σ denotes the sigmoid function. The initial states of the GRUs are set to zero vectors, i.e., $h_0 = 0$. Through the *sequence encoder* we obtain $H_A = \langle h_{A_1}, h_{A_2}, \ldots, h_{A_i}, \ldots, h_{A_N} \rangle$ for domain A, and $H_B = \langle h_{B_1}, h_{B_2}, \ldots, h_{B_j}, \ldots, h_{B_M} \rangle$ for domain B. We consider the last state as the in-domain representation, i.e., $h_A = h_{A_N}$ for domain A and $h_B = h_{B_M}$ for domain B. The in-domain representations are combined with the cross-domain representations (i.e., $h_{(A \to B)}$ or $h_{(B \to A)}$) to compute the final recommendation score. In the next two subsections, we describe two PSJNet instantiations that adopt different frameworks to learn the cross-domain representations.

3.3 PSJNet-I

In this subsection, we describe PSJNet-I in detail. PSJNet-I is a reformulation of π-Net [1] within the PSJNet framework, where we reformulate the shared account filter unit (SFU) and the cross-domain transfer unit (CTU) as a split-by-join unit. Fig. 2 provides an overview of PSJNet-I. PSJNet-I is a "Split-by-Join" framework; it gets the role-specific representations from the mixed user behavior and simultaneously joins them at each timestamp. Then the representations are transformed to another domain as cross-domain representations. PSJNet-I consists of three main components: (1) a *sequence encoder* (see Section 3.2), (2) a *split-by-join unit* (see this subsection), and (3) a *hybrid recommendation decoder* (see Section 3.5). The *sequence encoder* encodes the behavior sequences of each domain into high-dimensional hidden representations. The *split-by-join unit* takes each domain's representations as input and tries to first split the representations of specific user roles, and then joins and transforms them to another domain at each timestamp t. The *matching decoder* combines the information from both domains and generates a list of recommended items.

Under the shared account scenario, the behavior recorded under the same account is generated by different user roles. In practice, not all user roles that share the account are active in all domains. Besides, even though some user roles are active in the same domain, they may have different interests. For example, in the smart TV scenario, children may occupy the majority of the educational channel, while adults dominate the video TV channel.

The outputs H_A or H_B of the *sequence encoder* are the mixed representations of all user roles sharing the same account. To learn role-specific representations from these mixed representations, we propose a *split-by-join unit*, as shown in Fig. 3. The *split-by-join unit* can be applied bidirectionally from "domain A to domain B" and "domain B to domain A," meaning that the information is extracted from one domain and transferred to the other domain. Here, we take the "domain A to domain B" direction and achieving recommendations in domain B as an example. To learn role-specific representations, we need to know the number of user roles under each account, which is, unfortunately, unavailable in most cases. In this work, we assume that there are K latent roles (r_1, r_2, ..., r_k, ..., r_K) under each

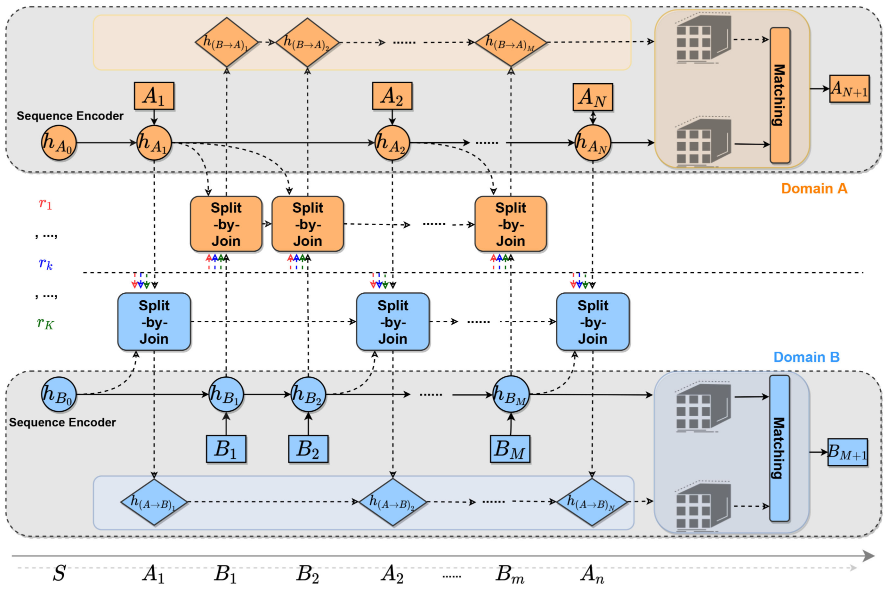
Sequence Encoder
Domain A
Domain B
Split-by-Join
Matching
r_1 , ..., r_k , ..., r_K
S A_1 B_1 B_2 A_2 ...... B_m A_n

Fig. 2. An overview of PSJNet-I. The orange and blue colors represent different domains. Red, purple and green represent different user roles. Section 3.3 contains a walkthrough of the model.

account. For example, in a family account, the user roles may correspond to child, male parent, female parent, etc. We first embed each latent role into $emb(r_k)$ $(1 \le k \le K)$, which represents and encodes the latent interests of each role. Then, we split the specific representation for role r_k at timestamp i in domain A with Eq. (3):

$$h^{r_k}_{A_i} = f^{r_k}_{A_i} \odot \widehat{h}^{r_k}_{A_i} + (1 - f^{r_k}_{A_i}) \odot h_{A_{i-1}\to B}, \tag{3}$$

where $\odot$ denotes element-wise multiplication. Mathematically, the representation $h^{r_k}_{A_i}$ is a combination of two representations $\widehat{h}^{r_k}_{A_i}$ and $h_{A_{i-1}\to B}$ balanced by $f^{r_k}_{A_i}$. A higher value of $f^{r_k}_{A_i}$ means that item A_i is more likely generated by r_k and we should pay more attention to r_k's related representation $\widehat{h}^{r_k}_{A_i}$. A lower value of lower $f^{r_k}_{A_i}$ means that item A_i might not be related to r_k and we should inherit more information from previous time steps.

Next, we introduce the definitions of the three parts of Eq. (3) one by one.

(a) We propose a gating mechanism to implement $f^{r_k}_{A_i}$ in Eq. (4):

$$f^{r_k}_{A_i} = \sigma(W_{f_A} \cdot h_{A_i} + W_{f_B} \cdot h_{B_j} + U_f \cdot h_{A_{i-1}\to B} + V_f \cdot emb(r_k) + b_f), \tag{4}$$

where $\cdot$ means matrix multiplication; W_{f_A}, W_{f_B}, U_f and V_f are the parameters; b_f is the bias term; h_{A_i} and h_{B_j} are the mixed representations of domain A and B at timestamp i and j, respectively. Note that B_j is the last item from domain B before A_i in the mixed sequence. $h_{A_{i-1}\to B}$ is the previous output, which will be explained later (under item (c)). After the sigmoid function σ, each value of $f^{r_k}_{A_i}$ falls into $(0, 1)$. Thus, the gating score $f^{r_k}_{A_i}$ controls the amount of information of r_k to transfer from domain A to domain B. It should be noted that each latent representation $emb(r_k)$ indicates the distribution of user roles with similar interests under each account, and it does not explicitly represents a specific user.

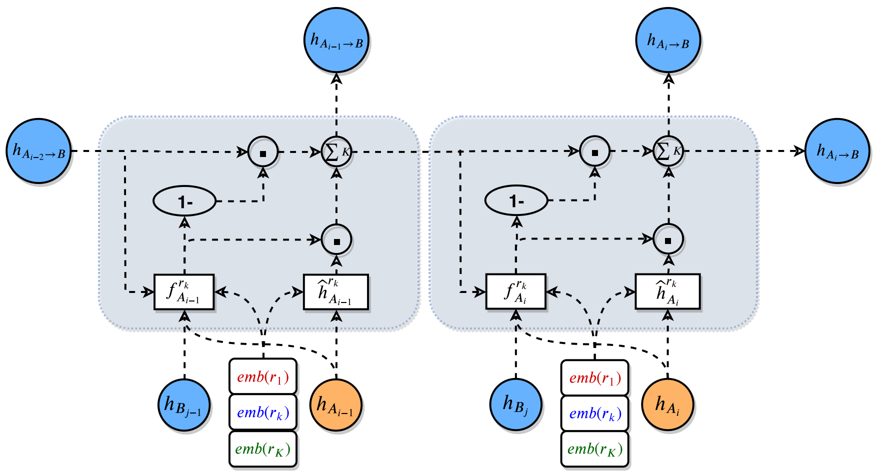

Fig. 3. The *split-by-join unit* illustrated while transforming information from domain A to domain B.

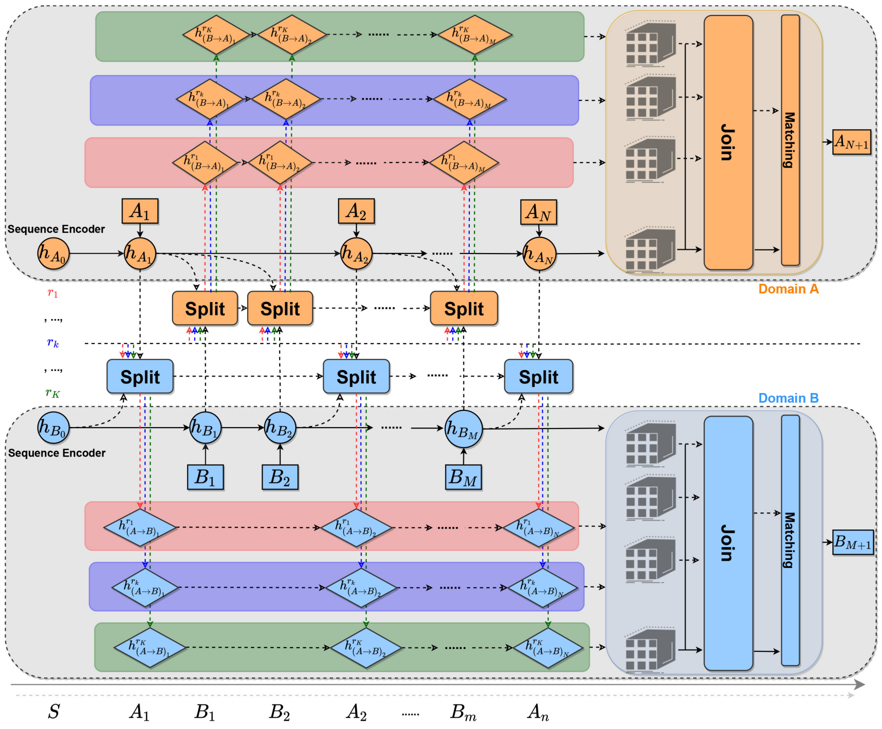

Fig. 4. An overview of PSJNet-II. As before, the orange and blue colors represent different domains. Red, purple and green represent different user roles. Section 3.4 contains a walkthrough of the model.

(b) $\widehat{h}^{r_k}_{A_i}$ is the candidate representation for r_k at timestamp i in domain A, which is computed based on the mixed representation h_{A_i}, the filtered previous output $h_{A_{i-1}\to B}$, and the user role r_k's latent interest $emb(r_k)$, as shown in Eq. (5):

$$\widehat{h}^{r_k}_{A_i} = \tanh(W_h \cdot h_{A_i} + U_h \cdot h_{A_{i-1}\to B} + V_h \cdot emb(r_k) + b_h), \tag{5}$$

where W_h, U_h and V_h are the parameters; b_h is the bias term.

(c) $h_{A_i\to B}$ is the final output of the cross-domain representation at timestamp i from domain A to domain B, which is calculated by joining each role-specific representation $h^{r_k}_{A_i}$

$$h_{A_i\to B} = \frac{1}{K}\sum_{k=1}^{K}\left(h^{r_k}_{A_i}\right). \tag{6}$$

Note that $h_{A_i\to B}$ is recurrently updated with Eqs. (3) and (6).

Using Eqs. (3) and (6), we obtain a sequence of representations $\langle h_{A_1\to B}, \ldots, h_{A_N\to B}\rangle$. We need to combine and transfer $\langle h_{A_1\to B}, \ldots, h_{A_N\to B}\rangle$ to domain B. We achieve this by employing another GRU to recurrently encode $h_{A_i\to B}$ at each timestep to obtain $h_{(A\to B)_i}$, as shown in Eq. (7):

$$h_{(A\to B)_i} = \mathrm{GRU}(h_{A_i\to B}, h_{(A\to B)_{i-1}}), \tag{7}$$

where $h_{A_i\to B}$ is the representation filtered from domain A; $h_{(A\to B)_{i-1}}$ is the previous transformed representation at timestamp $i-1$. The initial state is also set to zero vectors, i.e., $h_{(A\to B)_0} = 0$. We set the cross-domain representation from domain A to domain B (i.e., $h_{(A\to B)}$) as the last timestamp representation $h_{(A\to B)_N}$, where N is sequence length of domain A.

3.4 PSJNet-II

In this subsection, we describe PSJNet-II, our second solution for SAC-SR. Unlike PSJNet-I, PSJNet-II is a "Split-and-Join" framework, which means that it first splits role-specific representations from the mixed user behavior at each timestamp. Then the role-specific representations are transformed to another domain. Finally, it joins the role-specific representations as cross-domain representations. Fig. 4 provides an overview of PSJNet-II. PSJNet-II consists of four main components: (1) a *sequence encoder* (see Section 3.2), (2) a *split unit*, (3) a *join unit*, and (4) a *hybrid recommendation decoder* (see Section 3.5). PSJNet-II uses the same *sequence encoder* and *matching decoder* architectures as PSJNet-I. Please refer to Sections 3.2 and 3.5 for details of the *sequence encoder* and the *hybrid recommendation decoder*. In this subsection, we focus on the core modules (i.e., the *split unit* and *join unit*) of PSJNet-II.

3.4.1 *Split Unit*

The *split unit* is shown in Fig. 5. The differences with the *split-by-join unit* of PSJNet-I are marked in yellow. As with PSJNet-I, PSJNet-II also assumes that there are K latent roles under each account. We split the specific representation for role r_k at timestamp i in domain A with Eq. (8):

$$h^{r_k}_{A_i\to B} = f^{r_k}_{A_i} \odot \widehat{h}^{r_k}_{A_i} + f^{none}_{A_i} \odot h^{r_k}_{A_{i-1}\to B}, \tag{8}$$

where $f^{none}_{A_i}$ is a special gate that handles the case when none of the information from r_k at i (i.e., $\widehat{h}^{r_k}_{A_i}$) is useful and we should inherit more information from previous time steps, see Eq. (9):

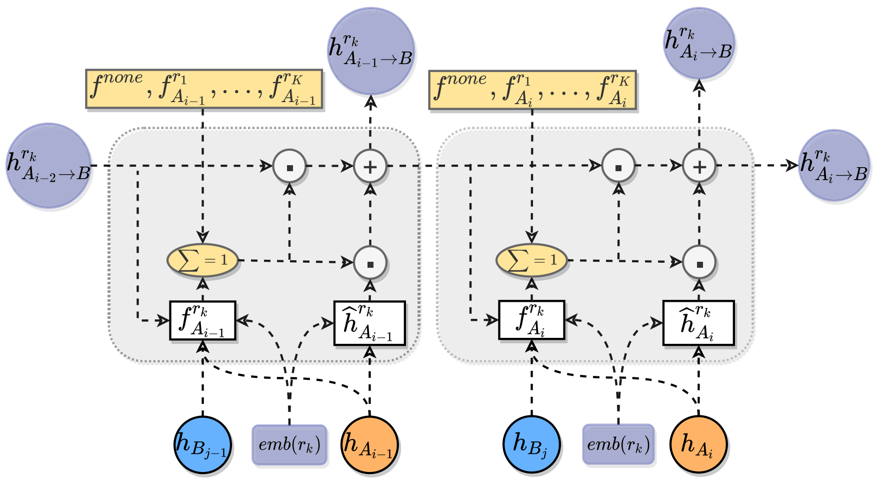
$f^{none}_{A_{i-1}}, f^{r_1}_{A_{i-1}}, \dots, f^{r_K}_{A_{i-1}}$
$h^{r_k}_{A_{i-1}\to B}$
$f^{none}_{A_i}, f^{r_1}_{A_i}, \dots, f^{r_K}_{A_i}$
$h^{r_k}_{A_i\to B}$
$h^{r_k}_{A_{i-2}\to B}$
$\sum = 1$
$f^{r_k}_{A_{i-1}}$
$\widehat{h}^{r_k}_{A_{i-1}}$
$f^{r_k}_{A_i}$
$\widehat{h}^{r_k}_{A_i}$
$h_{B_{j-1}}$
$emb(r_k)$
$h_{A_{i-1}}$
h_{B_j}
h_{A_i}

Fig. 5. The *split unit* for r_k illustrated while transforming information from domain A to domain B.

$$f^{none}_{A_i} = \sigma\Big(W_{f_A} \cdot h_{A_i} + W_{f_B} \cdot h_{B_j} + U_f \cdot h_{A_{i-1}\to B} + b_f\Big). \tag{9}$$

We add a normalization constraint to force the sum of $f^{r_k}_{A_i}$ and $f^{none}_{A_i}$ to 1:

$$f^{none}_{A_i} + \sum_{k=1}^{K} f^{r_k}_{A_i} = 1. \tag{10}$$

We use similar definitions of $f^{r_k}_{A_i}$ (Eq. (4)) and $\widehat{h}^{r_k}_{A_i}$ (Eq. (5)) as in PSJNet-I, except that $h_{A_{i-1}\to B}$ is replaced with $h^{r_k}_{A_{i-1}\to B}$. The differences from *split-by-join unit* are two-fold. First, $h^{r_k}_{A_i\to B}$ is not joined with respect to all roles. Second, instead of learning independent gates for different roles, we require that all gate values from all roles (and $f^{none}_{A_i}$) are summed to 1.

After Eq. (8), we get a sequence of representations $\langle h^{r_k}_{A_1\to B}, \dots, h^{r_k}_{A_n\to B}\rangle$ for each user role r_k. We combine and transfer $\langle h^{r_k}_{A_1\to B}, \dots, h^{r_k}_{A_n\to B}\rangle$ to domain B by employing another GRU, as shown in Eq. (11):

$$h^{r_k}_{(A\to B)_i} = \mathrm{GRU}(h^{r_k}_{A_i\to B}, h^{r_k}_{(A\to B)_{i-1}}), \tag{11}$$

where $h^{r_k}_{A_i\to B}$ is the representation filtered from domain A for role r_k.

3.4.2 Join Unit

The *join unit* is shown in Fig. 6. After the *split unit*, we obtain K sequences of transformed representations $\langle h^{r_k}_{(A\to B)_1}, \dots, h^{r_k}_{(A\to B)_N}\rangle$ from domain A to domain B. To join them, we first compute a similarity matrix $S^I \in \mathbb{R}^{M\times N}$ between the transformed representations and the in-domain representations $\langle h_{B_1}, \dots, h_{B_M}\rangle$ from domain B. Each similarity $S^I_{(i,j)}$ is computed with Eq. (12):

$$S^I_{(i,j)} = v_S{}^T \cdot (W_i \cdot h^{r_k}_{(A\to B)_i} + W_j \cdot h_{B_j}), \tag{12}$$

where $v_S{}^T$, W_i and W_j are parameters.

Then we pick the maximum similarity S^I_i between each $h^{r_k}_{(A\to B)_i}$ and all h_{B_j}. S^I_i signifies that $h^{r_k}_{(A\to B)_i}$ is more representative for role r_k in domain B because it has the closest similarity to a representation h_{B_j} in domain B:

$$S^I_i = \max_j S^I_{(i,j)}. \tag{13}$$

We normalize S^I_i with softmax afterwards. Then we obtain representations for each role r_k in Eq. (14):

$$h^{r_k}_{(A\to B)} = \sum_{i=1}^{N} (S^I_i h^{r_k}_{(A\to B)_i}). \tag{14}$$

Finally, we get the cross-domain representation $h_{(A\to B)}$ by joining $\langle h^{r_1}_{(A\to B)}, \dots, h^{r_K}_{(A\to B)}\rangle$ again with similar operations as in (12) and (14), but with a different similarity matrix $S^{II} \in \mathbb{R}^{M\times K}$. Note that S^{II} is computed between $\langle h^{r_1}_{(A\to B)}, \dots, h^{r_K}_{(A\to B)}\rangle$ and $\langle h_{B_1}, \dots, h_{B_M}\rangle$ this time.

There are two strengths of PSJNet-II compared to PSJNet-I. First, the normalization (see Eq. (10)) reduces the influence of some large gate values, thereby making the prediction more accurate. Second, the split-by-join unit of PSJNet-I uses the output of the last time step of GRU as the cross-domain representation from domain A to domain B. Information in the intermediate step is lost to some degree. However, in the join unit of PSJNet-II, the cross-domain representation from A to B undergoes more fine-grained calculations.

3.5 Hybrid Recommendation Decoder

The *hybrid recommendation decoder* integrates hybrid information from both domains A and B to evaluate the

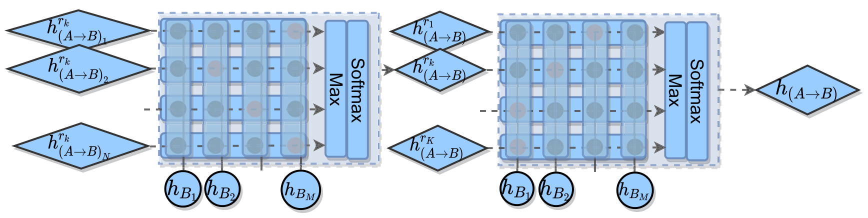

Fig. 6. The *join unit* illustrated while transforming information from domain A to domain B.

recommendation probabilities of the candidate items. Specifically, it first gets the hybrid representation by concatenating the representation h_B from domain B and the transformed representation $h_{(A\rightarrow B)}$ from domain A to domain B. Then, it evaluates the recommendation probability for each candidate item by calculating the matching of between the hybrid representation and the item-embedding matrix followed by a softmax function, as shown in Eq. (15):

$$P(B_{j+1}|S) = \mathrm{softmax}\Big(W_I \cdot \left[h_B, h_{(A\rightarrow B)}\right]^{\mathsf{T}} + b_I\Big), \tag{15}$$

where W_I is the embedding matrix of all items of domain B; b_I is the bias term.

3.6 Objective Function

We employ the negative log-likelihood loss function to train PSJNet in each domain as follows:

$$\begin{aligned} L_A(\theta) &= -\frac{1}{|\mathbb{S}|}\sum_{S\in\mathbb{S}}\sum_{A_i\in S}\log P(A_{i+1}\mid S) \\ L_B(\theta) &= -\frac{1}{|\mathbb{S}|}\sum_{S\in\mathbb{S}}\sum_{B_j\in S}\log P(B_{j+1}\mid S), \end{aligned} \tag{16}$$

where θ are all the parameters of our model PSJNet and $\mathbb{S}$ are the sequence instances in the training set. In the case of joint learning, the final loss is a linear combination of both losses:

$$L(\theta) = L_A(\theta) + L_B(\theta). \tag{17}$$

All parameters as well as the item embeddings are learned in an end-to-end back-propagation training way.

4 Experimental Setup

4.1 Research Questions

We seek to answer the following research questions in our experiments:

(RQ1) What is the performance of PSJNet-I and PSJNet-II on the SAC-SR task? Do they outperform the state-of-the-art methods in terms of Recall and MRR?

(RQ2) Which PSJNet variant is more effective in the SAC-SR task? PSJNet-I or PSJNet-II? What are the performances of different groups of methods, e.g., sequential and non-sequential recommendation methods?

(RQ3) What is the performance of PSJNet-I and PSJNet-II on different domains and different datasets? Do they improve the performance of both domains and datasets? Are the improvements equivalent?

4.2 Datasets

We need datasets that exhibit both shared-account and cross-domain characteristics to conduct experiments. To demonstrate the effectiveness of the proposed PSJNet model, we build and release two new datasets, HAmazon and HVIDEO, respectively. We build the HAmazon dataset by simulating shared account characteristics using previously released Amazon datasets. HVIDEO has previously been used in [1] but we release it with this paper. Details of the two datasets are as follows.

- HAmazon: He and McAuley [73] have released an Amazon product dataset that contains product reviews (ratings, text, helpfulness votes) and metadata (descriptions, category information, price, brand, and image features) from Amazon; it includes 142.8 million reviews spanning the period May 1996–July 2014. The data contains user behavior from multiple domains. In this paper, we use data from two Amazon domains. The M-domain contains *movie* watching and rating behavior of Amazon users. The B-domain covers *book* reading and rating behavior of Amazon users. We collect user-id, item-id, rating, and timestamp from the data and conduct some preprocessing. We first order the items by time under each account. Then, we merge records of the same item watched/read by the same user with an adjacent timestamp. The number of items in M-domain is less than that in B-domain. To balance the number of items in the hybrid interaction sequence, we only keep items whose frequency is larger than 5 in the M-domain and 10 in the B-domain.

  To satisfy cross-domain characteristics, we first find users whose behavior can be found in both the Amazon movie and book domains and then only keep users who have more than 10 records.

  To simulate shared account characteristics, we first split the data into 6 consecutive intervals, 1996–2000, 2001–2003, 2004–2006, 2007–2009, 2010–2012, and 2013–2015. We split the data in this way because we assume that the interaction behavior under the same account should be in the same time period. This is also a routine operation of many other works [74]. Then, we combine data from both domains by randomly merging 2, 3, or 4 users and their behavior in each interval as one shared account. Because each sequence has a lot of user behavior recorded over a long period of time, we split the sequences from each account into several small sequences with each containing watching/reading records within a year. We also require that each sequence contains at least 5 items from the M-domain and 2 items from the B-domain. The length of each sequence is between 4 and 60 with an average length of 31.29.

  For evaluation, we use the last watched/read item in each sequence for each domain as the ground truth respectively.

  We randomly select 75% of all data as the training set, 15% as the validation set, and the remaining 10% as the test set.

  The statistics of the final dataset are shown in Table 1. Note that although HAmazon can be used for experiments, it is not a SAC-SR dataset by nature. There are two shortcomings. First, the merged users do not naturally have the shared account characteristic. Second, the two domains are quite different and are not well correlated in content, which means that the items in different domains have little chance to reflect similar interests.
- HVIDEO: To facilitate future research for SAC-SR, we also release another dataset, HVIDEO, which exhibits shared-account and cross-domain characteristics by

TABLE 1
Statistics of the Datasets

HAmazon		HVIDEO	
M-domain		*V-domain*	
#Items	67,161	#Items	16,407
#Logs	4,406,924	#Logs	227,390
B-domain		*E-domain*	
#Items	126,547	#Items	3,380
#Logs	4,287,240	#Logs	177,758
#Overlapped-users	13,724	#Overlapped-users	13,714
#Sequences	289,160	#Sequences	134,349
#Training-sequences	204,477	#Training-sequences	102,182
#Validation-sequences	49,814	#Validation-sequences	18,966
#Test-sequences	34,869	#Test-sequences	13,201

nature. HVIDEO is a smart TV dataset that contains watching logs of 260k users from October 1st 2016 to June 30th 2017. The logs are collected on two platforms (the V-domain and the E-domain) from a well-known smart TV service provider. The V-domain contains family *video* watching behavior including TV series, movies, cartoons, talent shows and other programs. The E-domain covers online *educational* videos based on textbooks from elementary to high school, as well as instructional videos on sports, food, medical, etc.

On the two platforms, we gather user behavior, including which video is played, when a smart TV starts to play a video, and when it stops playing the video, and how long the video has been watched. Compared with previous datasets, HVIDEO contains rich and natural family behavior data generated in a natural shared account and cross-domain context. Therefore, it is very suitable for SAC-SR research.

We conduct some preprocessing on the dataset. We first filter out users who have less than 10 watching records and whose watching time is less than 300 seconds. Then, we merge records of the same item watched by the same user with an adjacent time less than 10 minutes. After that, we combine data from different domains together in chronological order which is grouped by the same account. Because each account has a lot of logs recorded in a long time, we split the logs from each account into several small sequences with each containing 30 watching records. This is a common preprocessing operation in SR tasks [37], [38], [75] And we require that the number of items in both domains must be greater than 5 in each sequence, which can ensure the sequences mix is high enough.

For evaluation, we use the last watched item in each sequence for each domain as the ground truth, respectively.

We randomly select 75% of all data as the training set, 15% as the validation set, and the remaining 10% as the test set. The statistics of the final dataset are shown in Table 1.

4.3 Baseline Methods

For our contrastive experiments, we consider baselines from four categories: traditional, sequential, shared account, and cross-domain recommendations.

4.3.1 Traditional Recommendations.

As traditional recommendation methods, we consider the following:

- POP: This method ranks items in the training set based on their popularity, and always recommends the most popular items. It is a very simple baseline, but it can perform well in certain domains and is frequently used as a baseline because of its simplicity [20].
- Item-KNN: The method computes a degree of item-to-item similarity that is defined as the number of co-occurrences of two items within sequences divided by the square root of the product of the number of sequences in which either item occurs. Items that are similar to the actual item will be recommended by this baseline. Regularization is included to avoid coincidental high similarities between rarely visited items [76].
- BPR-MF: This model is a commonly used matrix factorization method. This model cannot be applied directly to SRs, because new sequences do not have pre-computed feature vectors. Like [5], we apply it for sequential recommendations by representing a new sequence with the average latent factors of items that appeared in this sequence, i.e., we average the similarities of the feature vectors between a recommendable item and the items of the session so far.

4.3.2 Shared Account Recommendations.

There are some studies that explore shared account recommendations by first achieving user identification [50], [52]. However, they need extra information for user identification, e.g., some explicit ratings for movies or descriptions for some songs, even some textual descriptions for flight tickets, which is not available in our datasets. Therefore, we select a method that works on the IP-TV recommendation task that is similar to ours.

- VUI-KNN: This model encompasses an algorithm to decompose members in a composite account by mining different user interests over different time periods from logs [51]. The method first divides a day into time periods, so the logs of each account fall into the corresponding time period; logs in each time period are assumed to be generated by a virtual user that is represented by a 3-dimensional $\{account \times$

$item \times time\}$ vector. After that, cosine similarity is used to calculate similarity among virtual users, some of which are merged into a latent user. VUI deploys the User-KNN method to produce recommendations for these latent users.

4.3.3 *Cross-Domain Recommendations*

For cross-domain recommendations, we choose two baseline methods.

- NCF-MLP++: This model uses a deep learning-based process to learn the inner product of the traditional collaborative filtering by using a multilayer perceptron (MLP) [20]. We improve NCF-MLP by sharing the collaborative filtering in different domains. It is too time-consuming to rank all items with this method, because it needs to compute a score for each item one by one. We randomly sample four negative instances for each positive instance in the training process, and sample 3,000 negatives for each predicted target item in the test process, thus simplifying the task for this method.
- Conet: This is a neural transfer model across domains on the basis of a cross-stitch network [21], [69], where a neural collaborative filtering model [20] is employed to share information between domains.

4.3.4 *Sequential Recommendations*

Recently, a number of sequential recommendations methods have been proposed; RNN-based neural methods have outperformed traditional MC- or MDP-based methods. There are many RNN-based methods. In this paper, we consider two methods with somewhat similar architectures as PSJNet.

- GRU4REC: This model uses a GRU to encode sequential information. It uses a session-parallel mini-batch training process and employs ranking-based loss functions for learning the model [5].
- HGRU4REC: Quadrana *et al.* [9] propose this model based on RNNs which can deal with two cases: (1) user identifiers are present and propagate information from the previous sequence (user session) to the next, thus improving the recommendation accuracy, and (2) there are no past sessions (i.e., no user identifiers). The model is based on a hierarchical RNN, where the hidden state of a lower-level RNN at the end of one sequence is passed as input to a higher-level RNN, which is meant to predict a good initialization for the hidden state of the lower RNN for the next sequence.

4.4 Evaluation Metrics

Recommender systems can only recommend a limited number of items at a time. The item a user might pick should be amongst the first few in the ranked list [9], [77], [78]. Commonly used metrics are MRR@20 and Recall@20 [15], [79]. We also report MRR@5, Recall@5 and MRR@10, Recall@10.

- Recall: The primary evaluation metric is Recall, which measures the proportion of cases when the relevant item is amongst the top ranked items in all test cases. Recall does not consider the actual rank of the item as long as it is amongst the recommendation list. This accords with certain real-world scenarios where there is no highlighting of recommendations and the absolute order does not matter [5].
- MRR: Another used metric is MRR (Mean Reciprocal Rank), which is the average of reciprocal ranks of the relevant items. And the reciprocal rank is set to zero if the ground truth item is not in the list of recommendations. MRR takes the rank of the items into consideration, which is vital in settings where the order of recommendations matters. We choose MRR instead of other ranking metrics, because there is only one ground truth item for each recommendation; no ratings or grade levels are available.

For significance testing we use a paired t-test with $p < 0.05$. Significance testing is calculated against the results of the PSJNet-I.

4.5 Implementation Details

We set the item embedding size and GRU hidden state size to 90. We use dropout [80] with drop ratio $p = 0.8$. These settings are chosen with grid search on the validation set. For the latent user size K, we try different settings, an analysis of which can be found in Section 6.2. We initialize the model parameters randomly using the Xavier method [81]. We take Adam as our optimizing algorithm. For the hyper-parameters of the Adam optimizer, we set the learning rate $\alpha = 0.001$. We also apply gradient clipping [82] with range $[-5, 5]$ during training. To speed up the training and converge quickly, we use mini-batch size 64. We test the model performance on the validation set for every epoch. Both PSJNet-I and PSJNet-II are implemented in Tensorflow and trained on a GeForce GTX TitanX GPU.

5 Experimental Results

To answer RQ1, RQ2 and RQ3, we report the results of PSJNet compared with the baseline methods on the HAmazon and HVIDEO datasets, as shown in Tables 2 and 3, respectively. From the tables, we can see that both PSJNet-I and PSJNet-II outperform all baselines in terms of MRR and Recall for all reported values. Below, we discuss several insights we obtain from Tables 2 and 3 so as to answer our research questions.

5.1 Overall Performance on the SAC-SR Task (RQ1)

Both PSJNet variants significantly outperform all baselines and achieve the best results on all metrics, including strong baselines, i.e., GRU4REC and HGRU4REC. It is worth noting that although recent studies on SR propose many neural network models, we choose GRU4REC and HGRU4REC because GRU4REC and HGRU4REC have very similar architectures as PSJNet. And to obtain a fair comparison, we re-implement them within the same TensorFlow framework as we use for PSJNet.

Specifically, on the HVIDEO dataset, the largest increase of PSJNet-II over GRU4REC is 4.04% in terms of MRR@20, and 4.48% in terms of Recall@10 on the V-domain. On the E-domain, the increase is even larger with a 4.70% increase of PSJNet-II over GRU4REC in terms of MRR@20 and 13.03% increase of PSJNet-I over GRU4REC in terms of Recall@20. And the increase over HGRU4REC on the V-domain is 1.69%

TABLE 2
Experimental Results (%) on the HAmazon Dataset

Methods	M-domain recommendation						B-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
POP	0.36	0.44	0.49	0.73	1.32	2.02	0.14	0.19	0.22	0.42	0.78	1.22
Item-KNN	1.28	1.57	1.86	2.58	4.83	9.00	3.23	3.94	4.55	6.65	12.11	20.94
BPR-MF	2.90	3.00	3.06	3.90	4.65	5.50	0.88	0.92	0.96	1.23	1.50	2.15
VUI-KNN	–	–	–	–	–	–	–	–	–	–	–	–
NCF-MLP++	13.68	13.96	14.21	18.44	20.58	24.31	13.67	13.90	14.05	18.14	19.92	22.08
Conet	14.64	14.90	15.12	19.25	21.25	24.46	15.85	16.09	16.28	20.98	22.83	25.56
GRU4REC	82.01	82.08	82.11	83.10	83.61	84.06	81.34	81.41	81.44	82.77	83.32	83.76
HGRU4REC	83.07	83.12	83.14	84.24	84.65	84.91	82.15	82.26	82.31	83.46	84.30	84.91
PSJNet-I	83.91	83.94	83.95	**84.91**	**85.13**	**85.33**	84.93	84.93	84.93	**85.33**	85.36	**85.38**
PSJNet-II	**84.01**\rlap†	**84.04**\rlap†	**84.05**\rlap†	84.88	85.10	85.28	**85.10**\rlap†	**85.10**\rlap†	**85.11**\rlap†	85.32	**85.37**	**85.38**

Bold face *indicates the best result in terms of the corresponding metric. Significant improvements over the best baseline PSJNet-I results are marked with* † *(t-test,* $p < .05$*). To ensure a fair comparison, we re-implemented GRUE4REC and HGRU4REC in Tensorflow, just like PSJNet; the final results may be slightly different from the Theano version released by the authors. To obtain the results of NCF-MLP++ and Conet, we run the code released by Hu* et al. *[21]. VUI-KNN does not work on this dataset because it needs specific time in a day which is not available on the HAmazon dataset.*

TABLE 3
Experimental Results (%) on the HVIDEO Dataset

Methods	V-domain recommendation						E-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
POP	2.66	3.07	3.27	5.01	7.77	10.49	1.71	1.96	2.24	2.21	3.61	6.58
Item-KNN	4.43	4.16	2.93	10.48	16.49	23.93	2.11	2.39	2.90	3.01	5.77	12.11
BPR-MF	1.21	1.31	1.36	1.88	2.56	3.38	1.34	1.52	1.64	2.74	4.05	5.83
VUI-KNN	3.44	3.53	2.87	16.46	24.85	34.76	2.03	2.51	3.48	6.36	11.55	24.27
NCF-MLP++	16.25	17.25	17.90	26.10	33.61	43.04	3.92	4.57	5.14	7.36	12.27	20.84
Conet	21.25	22.61	23.28	32.94	43.07	52.72	5.01	5.63	6.21	9.26	14.07	22.71
GRU4REC	78.27	78.46	78.27	80.15	81.63	83.04	12.27	13.00	13.70	16.24	21.89	32.16
HGRU4REC	80.37	80.53	80.62	81.92	83.21	84.43	14.47	15.37	16.11	19.79	26.72	37.52
PSJNet-I	80.51	80.80	80.95	83.22	85.34	87.48	14.63	15.83	16.88	20.41	29.61	**45.19**
PSJNet-II	**81.97** †	**82.20** †	**82.32** †	**84.32** †	**86.11** †	**87.75** †	**16.63** †	**17.62** †	**18.46** †	**22.12** †	**29.64**	42.20

The same conventions are used as in Table 2.

and 3.45% (at most) in terms of MRR and Recall, respectively. On the E-domain, the increase is 2.29% and 7.67%, respectively. We believe that those performance improvements are due to the fact that PSJNet considers two important factors (shared-account and cross-domain) with its parallel modeling architecture and two main components for as part of its end-to-end recommendation model, namely the "split" and "join". With these three modules, PSJNet is able to model user interests more accurately by leveraging complementary information from both domains and improve recommendation performance in both domains. We will analyze the effects of the three modules in more depth in Section 6.1.

5.2 Comparing Two Versions of PSJNet With Different Groups of Methods (RQ2)

Generally, PSJNet-II outperforms PSJNet-I on both datasets. Specifically, PSJNet-II outperforms PSJNet-I in terms of most metrics on both domains on the HVIDEO dataset, especially for MRR@5 and Recall@5. Since both PSJNet-I and PSJNet-II adopt the parallel modeling architecture, we can conclude that the superiority of PSJNet-II over PSJNet-I mainly comes from the separate modeling of "split" and "join". We will show this in more depth in Section 6.1. However, the MRR values of PSJNet-II are the best but the Recall values are lower than those of PSJNet-I on the HAmazon. This is because we simulate the shared-account characteristic by merging 2, 3 or 4 users, in which the users under the same account do not have shared interests in most cases. Therefore, PSJNet-II does not have advantages over PSJNet-I on the HAmazon dataset, improving the recall value is difficult for PSJNet-II.

We can also see that RNN-based methods (e.g., GRU4REC, HGRU4REC, and our PSJNet) perform much better than traditional methods, which demonstrates that RNN-based methods are good at dealing with sequential information. They are able to learn better dense representations of the data through nonlinear modeling, which we assume is able to provide a higher level of abstraction. The shared account and cross-domain baselines (e.g., VUI-KNN, NCF-MLP++ and Conet) perform much worse than PSJNet. They also perform substantially worse than HGRU4REC.

TABLE 4
Analysis of the Contribution of the Parallel Modeling, Split Unit and Join Unit on the HAmazon Dataset

Methods	M-domain recommendation						B-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
PSJNet (-PSJ)	77.26	77.44	77.51	82.22	83.52	84.39	81.69	81.72	81.73	85.03	85.27	85.34
PSJNet-I (-SJ)	83.30	83.32	83.33	84.03	84.20	84.31	84.04	84.04	84.04	85.31	85.35	**85.38**
PSJNet-II (-S)	83.55	83.59	83.60	84.61	84.90	85.14	84.87	84.88	84.88	85.26	85.31	85.35
PSJNet-II (-J)	82.28	82.35	82.38	84.02	84.52	84.92	83.42	83.45	83.46	84.79	84.96	85.08
PSJNet-I	83.91	83.94	83.95	**84.91**	**85.13**	**85.33**	84.93	84.93	84.93	**85.33**	85.36	**85.38**
PSJNet-II	**84.01**	**84.04**	**84.05**	84.88	85.10	85.28	**85.10**	**85.10**	**85.11**	85.32	**85.37**	**85.38**

PSJNet (-PSJ) is PSJNet without parallel modeling, i.e., no cross-domain representations are used for recommendations. Without parallel modeling, both PSJNet-I and PSJNet-II become the same PSJNet (-PSJ). PSJNet-I (-SJ) is PSJNet-I without "split-by-join" unit. Because "split-by-join" is an indivisible unit, there is no PSJNet-I (-S) or PSJNet-I (-J). PSJNet-II (-S) is PSJNet-II without the "split" unit and PSJNet-II (-J) is PSJNet-II without the "join" unit.

This is because these shared account and cross-domain baselines ignore sequential information; VUI-KNN only considers the length of watching time, and NCF-MLP++ and Conet do not use any time information. Another reason is that NCF-MLP++ and Conet just map each overlapped account in both domains to the same latent space to calculate the user-item similarity. However, the existence of shared accounts makes it difficult to find the same latent space for different latent user roles under one account. Besides, VUI-KNN is not a deep learning method and it simply distinguishes user roles based on the fixed divided time periods in a day, which means it assumes there is only one family member in each time period. However, in the smart TV scenario, many people usually watch programs together [51]. This situation cannot be captured very well by VUI-KNN. And it requires the specific time of user behavior in a day in order to distinguish different user roles. That is why we cannot use it to obtain results on the HAmazon dataset because there is no such information. In contrast, PSJNet can extract components of each hidden user role according to their viewing records in another domain with the "split" module. The results of BPR-MF are lower than of POP, which indicates that most items users watched are very popular, which is in line with the phenomenon that people like to pursue popular items in the video and book recommendation scenarios.

5.3 Contrasting the Performance on Different Domains and Different Datasets (RQ3)

The Recall values of PSJNet on the HAmazon dataset are comparable on the two domains while the Recall values on the V-domain are higher than those on the E-domain on the HVIDEO dataset. This is also true for the other methods. We believe that this is because of data balance issues. On the HAmazon dataset, the data is generally balanced on two domains. Most accounts have an equal amount of data on both domains. This means that the models can learn pretty well with data from just one domain. Cross-domain information is not that important: the increase of PSJNet on the HAmazon dataset is relatively small. However, the situation is different on the HVIDEO dataset. Most accounts have much more data on the V-domain due to its content diversity; because of this, models can better learn user's viewing characteristics on the V-domain. Therefore, on the HAmazon dataset, the space for improvement on both domains is smaller than on the HVIDEO dataset.

Additionally, comparing PSJNet with the best baseline, HGRU4REC, we find that the largest increase on the E-domain is larger than on the V-domain. The largest increase in MRR is 1.69% on the V-domain and 2.29% on the E-domain. And for the Recall values, the largest increase is 3.45% on the V-domain, and 7.67% on the E-domain. This shows that the space for potential improvements on the V-domain is smaller than on the E-domain after considering shared account and cross-domain information.

Also, the increases in MRR and Recall are different on the two datasets. On the HAmazon dataset, there is no significant difference for both MRR and Recall from @5 to @20. This means that PSJNet can already predict the ground truth item within the top-5 for most cases. This is not true on the HVIDEO dataset, especially on the E-domain. Specifically, the largest increase is 2.25% for MRR from the top-5 to the top-20, and 24.78% for Recall.

6 Analysis

6.1 Ablation Study

In this subsection, we report on an ablation study to verify how well the parallel modeling schema, with the "split" and "join" units, improves the recommendation performance. The results are shown in Tables 4 and 5. PSJNet (-PSJ) is the PSJNet-I or PSJNet-II without all three parts, which degenerates into GRU4REC except that PSJNet (-PSJ) is jointly trained on two domains. PSJNet-I (-SJ) is PSJNet-I without "split-by-join" unit. PSJNet-II (-S) is PSJNet-II without the "split" unit and PSJNet-II (-J) is PSJNet-II without the "join" unit (i.e., replacing the "join" unit by summing up the outputs from the "split" unit). We can draw the following conclusions from the results.

First, almost all the best results are almost all from PSJNet-I and PSJNet-II, which demonstrates the effectiveness of combining all three parts. The three parts bring around 7% (MRR) and 1%–3% (Recall) improvements on the M-domain of HAmazon, and around 4% (MRR) and 4%–10% (Recall) on both domains of HVIDEO. Additionally, we see that PSJNet (-PSJ) gets the lowest performance amongst these methods, while it still outperforms most of the baselines listed in Tables 2 and 3. In summary, then, combining information from an auxiliary domain is useful. The MRR improvements are larger on HAmazon while the Recall improvements are larger on HVIDEO. This is due to the different characteristics of different domains. Take the two domains in HVIDEO for example. Almost all members

TABLE 5
Analysis of the Contribution of the Parallel Modeling, Split Unit and Join Unit on the HVIDEO Dataset

Methods	V-domain recommendation						E-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
PSJNet (-PSJ)	78.02	78.17	78.28	80.13	81.34	82.93	12.69	13.43	14.05	16.54	22.29	31.45
PSJNet-I (-SJ)	78.59	78.85	78.97	81.71	83.58	85.33	16.35	17.04	17.59	20.97	26.29	34.44
PSJNet-II (-S)	81.61	81.85	81.99	83.93	85.73	87.71	15.94	17.01	17.84	20.96	29.18	41.38
PSJNet-II (-J)	81.76	81.98	82.12	84.20	85.80	**87.77**	16.43	17.48	**18.46**	21.92	**29.96**	44.30
PSJNet-I	80.51	80.80	80.95	83.22	85.34	87.48	14.63	15.83	16.88	20.41	29.61	**45.19**
PSJNet-II	**81.97**	**82.20**	**82.32**	**84.32**	**86.11**	87.75	**16.63**	**17.62**	**18.46**	**22.12**	29.64	42.20

We use the same conventions as in Table 4.

TABLE 6
Analysis of the Hyperparameter K on the HAmazon Dataset

K values	M-domain recommendation						B-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
						PSJNet-I						
1	82.45	82.52	82.54	84.23	84.69	85.07	84.72	84.73	84.73	85.29	85.35	85.38
2	83.35	83.40	83.41	84.66	85.02	85.18	84.74	84.75	84.75	85.30	85.25	85.37
3	83.65	83.68	83.70	84.81	85.08	85.30	84.89	84.89	84.89	85.32	85.35	85.38
4	**83.91**	**83.94**	**83.95**	**84.91**	**85.13**	**85.33**	84.93	84.93	84.93	**85.33**	**85.40**	85.38
5	83.73	83.76	83.78	84.82	85.08	85.32	**84.94**	**84.94**	**84.94**	**85.33**	85.38	**85.39**
						PSJNet-II						
1	84.33	**84.36**	**84.37**	**85.01**	**85.19**	**85.32**	85.09	85.10	85.10	85.32	85.36	85.39
2	**84.08**	84.12	84.13	84.92	85.15	85.30	85.13	85.13	85.13	**85.33**	85.36	**85.40**
3	84.03	84.06	84.07	84.92	85.12	85.29	**85.16**	**85.16**	**85.16**	**85.33**	85.35	85.37
4	84.01	84.04	84.05	84.88	85.10	85.28	85.10	85.10	85.11	85.32	**85.37**	85.38
5	82.34	82.42	82.44	84.06	84.63	84.99	84.67	84.68	84.69	85.23	85.30	85.37

have viewing records in the V-domain. However, items on the E-domain are mostly educational programs, so children take up a large proportion, and their educational interests are relatively fixed. As a result, the information extracted from the V-domain mostly belongs to children, which is less helpful because we already have enough data on the E-domain to learn such features in most cases.

Second, generally parallel modeling brings the most improvements followed by the "split" and "join" units. Specifically, PSJNet-I achieves around 5% (MRR) and 2% (Recall) improvements on the M-domain of HAmazon with the parallel modeling while further improvements with the "split-by-join" unit are just around 0.6% (MRR) and 1% (Recall). Similar results can be found on the B-domain of HAmazon and E-domain of HVIDEO. We believe this is because the model is already able to leverage information from both domains to achieve recommendations with the parallel modeling schema. It is further improved by taking other factors, e.g., shared-account characteristics, into account in order to better leverage the cross-domain information. This is why the "split" and "join" units are able to further improve the results over the parallel modeling schema. An exception is that the "split" and "join" units achieve more improvements than the parallel modeling on the V-domain of HVIDEO for PSJNet-I. We think the reason is that PSJNet-I (-SJ) cannot effectively use the cross-domain information without the "split-by-join" unit, while PSJNet-II (-S) is better because the function of "split" unit is replaced by the "join" unit to some extent. The same is true for PSJNet-II (-J). This could be verified by the fact that both PSJNet-I and PSJNet-II get big improvements with both units than with neither, but the improvements are smaller than with one unit for PSJNet-II.

Third, the "split" unit is generally more effective than the "join" unit for PSJNet-II as we find that the gap between PSJNet-II and PSJNet-II (-J) is smaller than between PSJNet-II and PSJNet-II (-S). On the one hand, this shows that the "split" unit plays a more important role which addresses the challenge raised by shared accounts, i.e., filtering out information of some user roles that might be useful for another domain from the mixed user behaviors. On the other hand, the results also show that the current "join" unit is not effective enough as directly summing up the outputs from the "split" unit also achieves competitive performance, and/or the improvement space of the "join" unit is limited.

6.2 Influence of the Hyperparameter K

Both PSJNet-I and PSJNet-II introduce a hyperparameter K in the "split" unit which corresponds to the number of latent user roles. We carry out experiments to study how setting K affects the recommendation performance of PSJNet on both datasets, and whether the best K is the same under all situations and accords with reality. Taking into account common sizes of families, we consider $K = 1, \ldots, 5$, and compare different values of K while keeping other settings unchanged. The results are shown in Tables 6 and 7.

First, we see that the best values in terms of MRR and Recall are achieved when $K = 4, 5$ for PSJNet-I and $K = 1, 3$ for PSJNet-II. We believe that this is because PSJNet-II

TABLE 7
Analysis of the Hyperparameter K on the HVIDEO Dataset

K values	V-domain recommendation						E-domain recommendation					
	MRR			Recall			MRR			Recall		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
						PSJNet-I						
1	80.19	80.50	80.66	82.85	85.15	87.40	13.92	15.06	16.10	19.76	28.74	43.98
2	80.48	80.75	80.91	83.08	85.06	87.31	14.29	15.47	16.54	19.83	28.96	44.77
3	80.53	80.79	80.93	**83.34**	85.31	87.31	14.45	15.54	16.64	20.23	28.61	44.64
4	80.51	80.80	80.95	83.22	**85.34**	**87.48**	**14.63**	**15.83**	**16.88**	20.41	**29.61**	**45.19**
5	**80.60**	**80.86**	**81.02**	83.25	85.19	87.47	14.59	15.71	16.75	**20.42**	28.97	44.38
						PSJNet-II						
1	81.93	82.18	82.32	**84.33**	**86.17**	**88.21**	16.17	17.18	18.13	21.42	29.23	**43.29**
2	81.80	82.04	82.17	84.26	86.05	87.90	16.62	17.67	18.55	21.60	29.60	42.63
3	81.86	82.08	82.20	84.14	85.80	87.53	**16.90**	**17.94**	**18.77**	**22.42**	**30.36**	42.51
4	**81.97**	**82.20**	**82.32**	84.32	86.11	87.75	16.63	17.62	18.46	22.12	29.64	42.20
5	81.78	82.02	82.14	83.99	85.67	87.68	16.78	17.84	18.66	22.01	30.07	42.13

models different user roles separately. Similar users may have the similar user role, thus PSJNet-II performs better when the number of k is lower, which demonstrates the different trends of PSJNet-I and PSJNet-II. This is consistent with the size of modern families on HVIDEO and the simulation settings on HAmazon. For PSJNet-I, the lowest MRR and Recall values are achieved when $K = 1$. But for PSJNet-II, the gap between the best and worst performances is much smaller, which indicates that PSJNet-II is less sensitive to K than PSJNet-I.

Second, the distribution of the best results of PSJNet-I on two different datasets is consistent, i.e., the best K values are basically the same, and so is PSJNet-II. But PSJNet-I and PSJNet-II have different best results on the same dataset. On the one hand, this demonstrates the performance stability of both PSJNet-I and PSJNet-II. On the other hand, this is also a clue that both models identify K as the potential user roles under each account, which verifies our assumption.

Third, although K can affect the recommendation performance, the influence is limited. As we can see that the largest gaps between the best and worst performances are 1.94% (MRR) and 0.56% (Recall) on HAmazon, 0.78% (MRR) and 1.21% (Recall) on HVIDEO. This is because even if $K = 1, 2$, our models still consider the information of all members except that some members are modeled as a single latent user role.

7 Conclusion and Future Work

We have studied the task of SAC-SR and proposed an extension to our previous work [1]. We have generalized over the previous proposal (π-Net) with a more general framework that allows us to come up with a better performing model. Under this framework, we have reformulated π-Net as PSJNet-I and proposed a new instantiation, PSJNet-II, with different split-join schemes. Experimental results demonstrate that PSJNet outperforms state-of-the-art methods in terms of MRR and Recall. We have also conducted extensive analysis experiments to show the effectiveness of the two PSJNet variants.

A limitation of PSJNet is that it works better only when we have shared information in two domains that are complementary to each other. Otherwise, PSJNet only achieves comparable performance with state-of-the-art methods for shared account and/or cross-domain recommendations.

As to future work, PSJNet can be advanced in several directions. First, we assume the same number of latent user roles under each account in this study. This can be further improved by automatically detecting the number of user roles, e.g., adaptively setting the number of family members in smart TV scenarios. Second, we have focused on the architecture of PSJNet and have not explored alternative choices for some of its main ingredients (e.g., encoders, decoders and loss functions). It would be interesting to see whether alternative choices will further improve the performance of PSJNet. Third, it is interesting to see whether it will further improve the performance by explicitly modeling the number of users under the same account. Unfortunately, we cannot find any datasets that exhibit such characteristics, so we leave this for future work. Fourth, explainability is seen as an important challenge for deep learning at present [83], [84]. It is interesting to see how effective explanations can be produced for different stakeholders in the complex domain of SAC-SR [85]. Fifth, we consider two domains in this work. However, we think it is completely practicable to extend this work to multiple domains by adjusting the "split" and "join" units slightly, e.g., one "split" for each pair of domains.

Code and Data

The code used to run the experiments in this paper is available at https://bitbucket.org/Catherine_Ma/sequentialrec/src/master/tois-PsiNet/code/. The datasets released in this paper are shared at https://bitbucket.org/Catherine_Ma/sequentialrec/src/master/tois-PsiNet/datasets/.

Acknowledgments

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors. This paper is a substantially extended version of [1]. The additions are three-fold. First, we unify the parallel modeling framework introduced in [1] into the PSJNet architecture introduced in this paper and propose a new model PSJNet-II that improves the performance over previous proposals (π-Net in [1] corresponds to PSJNet-I in this

paper). Second, we build a new dataset for shared account cross-domain sequential recommendation by simulating shared account characteristics on a public dataset. Third, we carry out more experiments to test PSJNet-I and PSJNet-II. More than half of the experiments reported in this paper were not in [1] and all relevant result tables and figures are either new additions to the article or report new results. Wenchao Sun and Muyang Ma are Co-first author.

References

[1] M. Ma, P. Ren, Y. Lin, Z. Chen, J. Ma, and M. de Rijke, "π-Net: A parallel information-sharing network for cross-domain shared-account sequential recommendations," in *Proc. 42st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 685–694.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] Y. Yan, M. Tan, I. W. Tsang, Y. Yang, Q. Shi, and C. Zhang, "Fast and low memory cost matrix factorization: Algorithm, analysis, and case study," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 2, pp. 288–301, Feb. 2020.

[5] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016.

[6] R. He and J. McAuley, "Fusing similarity models with Markov chains for sparse sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 191–200.

[7] Z. Cheng, J. Shen, L. Zhu, M. S. Kankanhalli, and L. Nie, "Exploiting music play sequence for music recommendation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3654–3660.

[8] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, 2005.

[9] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 130–137.

[10] A. Zimdars, D. M. Chickering, and C. Meek, "Using temporal data for making recommendations," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, 2001, pp. 580–588.

[11] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 66:1–66:36, 2018.

[12] M. Ma *et al.*, "Improving transformer-based sequential recommenders through preference editing," 2021, *arXiv:2106.12120*.

[13] W. Chen, P. Ren, F. Cai, F. Sun, and M. de Rijke, "Multi-interest diversification for end-to-end sequential recommendation," *ACM Trans. Inf. Syst.*, vol. 40, no. 1, Sep. 2021. [Online]. Available: https://doi.org/10.1145/3475768

[14] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 1053–1058.

[15] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke, "RepeatNet: A repeat aware neural recommendation machine for session-based recommendation," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 4806–4813.

[16] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 185–194.

[17] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 595–606.

[18] K. Verstrepen and B. Goethals, "Top-N recommendation for shared accounts," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 59–66.

[19] Y. Zhao, J. Cao, and Y. Tan, "Passenger prediction in shared accounts for flight service recommendation," in *Proc. 10th Int. Conf. Asia-Pacific Serv. Comput.*, 2016, pp. 159–172.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[21] G. Hu, Y. Zhang, and Q. Yang, "CoNet: Collaborative cross networks for cross-domain recommendation," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 667–676.

[22] F. Zhuang, Y. Zhou, F. Zhang, X. Ao, X. Xie, and Q. He, "Sequential transfer learning: Cross-domain novelty seeking trait mining for recommendation," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 881–882.

[23] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for next basket recommendation," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 403–412.

[24] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 811–820.

[25] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 714–722.

[26] Z. Lu, E. Zhong, L. Zhao, E. W. Xiang, W. Pan, and Q. Yang, "Selective transfer learning for cross domain recommendation," in *Proc. SIAM Int. Conf. Data Mining*, 2013, pp. 641–649.

[27] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 729–732.

[28] H. Fang, D. Zhang, Y. Shu, and G. Guo, "Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations," *ACM Trans. Inf. Syst.*, vol. 39, no. 1, Nov. 2020, Art. no. 10.

[29] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, "A survey on large-scale machine learning," *IEEE Trans. Knowl. Data Eng.*, 2020.

[30] P. Sun, L. Wu, and M. Wang, *Attentive Recurrent Social Recommendation*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 185–194. [Online]. Available: https://doi.org/10.1145/3209978.3210023

[31] C. Chen, D. Li, J. Yan, and X. Yang, "Modeling dynamic user preference via dictionary learning for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, 2021.

[32] J. Wu, R. Cai, and H. Wang, "Déjà vu: A contextualized temporal attention mechanism for sequential recommendation," in *Proc. Web Conf.*, 2020, pp. 2199–2209.

[33] Z. Wang *et al.*, "Counterfactual data-augmented sequential recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 347–356.

[34] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 17–22.

[35] Q. Cui, S. Wu, Q. Liu, W. Zhong, and L. Wang, "MV-RNN: A multi-view recurrent neural network for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 2, pp. 317–331, Feb. 2020.

[36] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 565–573.

[37] W.-C. Kang and J. Mcauley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 197–206.

[38] J. Li, Y. Wang, and J. McAuley, "Time interval aware self-attention for sequential recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 322–330.

[39] A. Luo *et al.*, "Collaborative self-attention network for session-based recommendation," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 2591–2597.

[40] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, *Next-Item Recommendation with Sequential Hypergraphs*. New York, NY, USA: Assoc. Comput. Machinery, 2020, pp. 1101–1110.

[41] H. Li, X. Wang, Z. Zhang, J. Ma, P. Cui, and W. Zhu, "Intention-aware sequential recommendation with structured intent transition," *IEEE Trans. Know. Data Eng.*, 2021.

[42] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020. [Online]. Available: http://dx.doi.org/10.1145/3397271.3401137

[43] Y. Zhang *et al.*, "How to retrain recommender system?," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020. [Online]. Available: http://dx.doi.org/10.1145/3397271.3401167

[44] J. Wu *et al.*, "Self-supervised graph learning for recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021. [Online]. Available: http://dx.doi.org/10.1145/3404835.3462862

[45] X. Wang *et al.*, *Learning Intents behind Interactions With Knowledge Graph for Recommendation*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 878–887.

[46] X. Chen *et al.*, "Sequential recommendation with user memory networks," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 108–116.

[47] M. Wang, P. Ren, L. Mei, Z. Chen, M. Jun, and M. de Rijke, "A collaborative session-based recommendation approach with parallel memory modules," in *Proc. 42st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 345–354.

[48] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 306–310.

[49] A. Zhang, N. Fawaz, S. Ioannidis, and A. Montanari, "Guess who rated this movie: Identifying users through subspace clustering," in *Proc. 28th Conf. Uncertainty Artif. Intell.*, 2012, pp. 944–953.

[50] P. Bajaj and S. Shekhar, "Experience individualization on online tv platforms through persona-based account decomposition," in *Proc. 24th ACM Int. Conf. Multimedia*, 2016, pp. 252–256.

[51] Z. Wang, Y. Yang, L. He, and J. Gu, "User identification within a shared account: Improving IP-TV recommender performance," in *Proc. 18th East Eur. Conf. Advances Databases Inf. Syst.*, 2014, pp. 219–233.

[52] J.-Y. Jiang, C.-T. Li, Y. Chen, and W. Wang, "Identifying users behind shared accounts in online streaming services," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 65–74.

[53] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[54] F. Abel, E. Herder, G.-J. Houben, N. Henze, and D. Krause, "Cross-system user modeling and personalization on the social web," *User Model. User-Adapted Interact.*, vol. 23, no. 2, pp. 169–209, 2013.

[55] S. Berkovsky, T. Kuflik, and F. Ricci, "Mediation of user models for enhanced personalization in recommender systems," *User Model. User-Adapted Interact.*, vol. 18, no. 3, pp. 245–286, 2008.

[56] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? Cross-domain collaborative filtering for sparsity reduction," in *Proc. 17th Int. Joint Conf. Artif. Intell.*, 2009, pp. 2052–2057.

[57] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 230–235.

[58] A. Farseev, I. Samborskii, A. Filchenkov, and T.-S. Chua, "Cross-domain recommendation via clustering on multi-layer graphs," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 195–204.

[59] I. Fern ández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci, "Cross-domain recommender systems: A survey of the state of the art," in *Proc. Spanish Conf. Inf. Retrieval*, 2012, pp. 1–12.

[60] J. Tang, S. Wu, J. Sun, and H. Su, "Cross-domain collaboration recommendation," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 1285–1293.

[61] Q. Do, W. Liu, J. Fan, and D. Tao, "Unveiling hidden implicit similarities for cross-domain recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 1, pp. 302–315, Jan. 2021.

[62] L. Chen, J. Zheng, M. Gao, A. Zhou, W. Zeng, and H. Chen, "TLRec: Transfer learning for cross-domain recommendation," in *Proc. IEEE Int. Conf. Big Knowl.*, 2017, pp. 167–172.

[63] Z. Liu, J. Tian, L. Zhao, and Y. Zhang, "Attentive-feature transfer based on mapping for cross-domain recommendation," in *Proc. Int. Conf. Data Mining Workshops*, 2020, pp. 151–158.

[64] T.-N. Doan and S. Sahebi, "TransCrossCF: Transition-based cross-domain collaborative filtering," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl.*, 2020, pp. 320–327.

[65] H. Zhang, X. Kong, and Y. Zhang, "Selective knowledge transfer for cross-domain collaborative recommendation," *IEEE Access*, vol. 9, pp. 48039–48051, 2021.

[66] K. D. Onal *et al.*, "Neural information retrieval: At the end of the early years," *Inf. Retrieval J.*, vol. 21, no. 2/3, pp. 111–182, Jun. 2018.

[67] C. Gao *et al.*, "Cross-domain recommendation without sharing user-relevant data," in *Proc. World Wide Web Conf.*, 2019, pp. 491–502.

[68] M. Ma *et al.*, "Mixed information flow for cross-domain sequential recommendations," 2020, *arXiv:2012.00485*.

[69] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3994–4003.

[70] H. Chen, H. Yin, X. Sun, T. Chen, B. Gabrys, and K. Musial, *Multi-Level Graph Convolutional Networks for Cross-Platform Anchor Link Prediction*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1503–1511.

[71] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," 2020, *arXiv:2012.06852*.

[72] L. Guo, L. Tang, T. Chen, L. Zhu, Q. V. H. Nguyen, and H. Yin, "DA-GCN: A domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation," 2021, *arXiv:2105.03300*.

[73] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 507–517.

[74] W. Wang *et al.* "Group-aware long- and short-term graph representation learning for sequential group recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Devel. Inform. Retrieval*, 2020, pp. 1449–1458.

[75] Z. Shuai, T. Yi, L. Yao, and A. Sun, "Next item recommendation with self-attention," 2018, *arXiv:1808.06414*.

[76] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[77] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: Rating prediction with ratings and reviews," in *Proc. World Wide Web Conf.*, 2018, pp. 639–648.

[78] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial personalized ranking for recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 355–364.

[79] L. Mei, P. Ren, Z. Chen, L. Nie, J. Ma, and J.-Y. Nie, "An attentive interaction network for context-aware recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 157–166.

[80] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[81] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist. Soc. Artif. Intell. Statist.*, 2010, pp. 249–256.

[82] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Int. Conf. Mach. Learn.*, 2013, pp. III-1310–III-1318.

[83] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, "Counterfactual explainable recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1784–1793.

[84] P. Sun, L. Wu, K. Zhang, Y. Fu, R. Hong, and M. Wang, "Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation," in *Proc. Web Conf.*, 2020, pp. 837–847.

[85] A. Lucic *et al.*, "A multistakeholder approach towards evaluating AI transparency mechanisms," in *Proc. ACM CHI Workshop Operationalizing Hum.-Centered Perspectives Explainable AI*, 2021.

Wenchao Sun is currently working toward the PhD degree at Shandong University, China. His research interests include information retrieval, recommender systems, and deep learning.

Muyang Ma is currently working toward the PhD degree at Shandong University, China. She is a student member of China Computer Federation. Her research interests include recommendation system.

Pengjie Ren is currently a professor at Shandong University, China. His research interests include summarization systems, search systems, recommender systems and dialogue systems. He has published papers on conferences and journals, including but not limited to ACM SIGIR, The Web Conference, ACL, ACM CCS, AAAI, EMNLP, COLING, ACM CIKM, the *ACM Transactions on Information Systems*, the *IEEE Transactions on Knowledge and Data Engineering*, the *ACM Transactions on Knowledge Discovery from Data*, the *Artificial Intelligence*, etc.

Yujie Lin is currently working toward the PhD degree at Shandong University, China. His research interests include information retrieval, recommender systems, and deep learning. He has published several papers on the *IEEE Transactions on Knowledge and Data Engineering*, WWW, SIGIR and so on.

Zhumin Chen is currently a professor and PhD supervisor in Shandong University, China. He is a senior member of China Computer Federation. His research interests include natural language processing, big data processing, recommendation systems, etc.

Zhaochun Ren is currently a professor and PhD supervisor in Shandong University, China. His research interests include information retrieval and natural language processing, especially recommendation systems, knowledge graph mining, question answering and dialogue systems, and social media analysis. He has published papers on conferences and journals, including but not limited to SIGIR, ACL, KDD, WSDM, CIKM, the *ACM Transactions on Information Systems*, the *IEEE Transactions on Knowledge and Data Engineering*, AAAI, IJCAI, WWW, etc.

Jun Ma is currently a professor and PhD supervisor in Shandong University, China. He is a senior member of China Computer Federation. His research interests include natural language processing, big data processing, recommendation systems, etc.

Maarten de Rijke is currently a distinguished university professor of artificial intelligence and information retrieval at the University of Amsterdam, the Netherlands. He is also the scientific director of the National Innovation Center for Artificial Intelligence (ICAI). His research is focused on designing trustworthy technology to connect people to information, particularly search engines, recommender systems, and conversational assistants. His work targets two key questions: How can we establish extrinsic trust in information retrieval systems, that is, establish verifiable guarantees on their behavior? And how can we create intrinsic trust in information retrieval systems, that is, align their reasoning process with human expectations?