applied sciences

Article

Self-Attentive Subset Learning over a Set-Based Preference in Recommendation

Kunjia Liu, Yifan Chen *, Jiuyang Tang, Hongbin Huang and Lihua Liu

Laboratory for Big Data and Decision, National University of Defense Technology, Changsha 410073, China
* Correspondence: yfchen@nudt.edu.cn

Abstract: Recommender systems that learn user preference from item-level feedback (provided to individual items) have been extensively studied. Considering the risk of privacy exposure, learning from set-level feedback (provided to sets of items) has been demonstrated to be a better way, since set-level feedback reveals user preferences while to some extent, hiding his/her privacy. Since only set-level feedback is provided as a supervision signal, different methods are being investigated to build connections between set-based preferences and item-based preferences. However, they overlook the complexity of user behavior in real-world applications. Instead, we observe that users' set-level preference can be better modeled based on a subset of items in the original set. To this end, we propose to tackle the problem of identifying subsets from sets of items for set-based preference learning. We propose a policy network to explicitly learn a personalized subset selection strategy for users. Given the complex correlation between items in the set-rating process, we introduce a self-attention module to make sure all set members are considered in subset selecting process. Furthermore, we introduce gumble softmax to avoid gradient vanishing caused by binary selection in model learning. Finally the selected items are aggregated by user-specific personalized positional weights. Empirical evaluation with real-world datasets verifies the superiority of the proposed model over the state-of-the-art.

Keywords: recommender systems; set-based preference learning; subset selection; user behav-ior modeling

check for updates

Citation: Liu, K.; Chen, Y.; Tang, J.; Huang, H.; Liu, L. Self-Attentive Subset Learning over a Set-Based Preference in Recommendation. *Appl. Sci.* **2023**, *13*, 1683. https://doi.org/10.3390/app13031683

Academic Editor: Giacomo Fiumara

Received: 5 January 2023
Revised: 25 January 2023
Accepted: 26 January 2023
Published: 28 January 2023

1. Introduction

Ubiquitous recommender systems provide effective solutions to improve users' experiences with online service platforms. Their main goal is to infer users' interests through their historical feedback information. Among the recommendation models applied, collaborative filtering (CF) approaches have demonstrated effective performance [1]. Most CF methods directly rely on the item-level feedback of users [2,3], which causes privacy concerns, since the users' preferences towards individual items are private opinions. Moreover, this privacy risk potentially inhibits users from providing feedback in the first place. Thus, avoiding direct item-level preference exposure is reasonable in recommender systems [4].

We refer to the problem of understanding the user's preference for individual items by learning from set-level feedback as *set-based preference learning* (SPL).The set-based feedback reveals users' preferences and naturally provides information hiding by blurring interactions with individual items. Thus, users are more willing to provide set-based feedback than item-level preferences. However, inferring item preference from set-based feedback is nontrivial. Since only set-based preference is provided, there is no direct supervision signal for item-level rating prediction. Therefore, the major challenge of SPL is *how to accurately infer a user's preferences for individual items based on set-level feedback.*

Since only set ratings are provided as supervision signals, the item-level predictions should be aggregated as set-based preferences before being updated. Although a set consists of several items, set-based feedback cannot just be modeled by adding up item-level feedback, because it is also influenced by the implicit interplay between items [5]. For example, some users tend to give positive feedback as long as the set contains something that piques their interest, whereas some users are inclined to underrate a set if it contains undesirable items. Moreover, users might not follow the same rating patterns when facing different sets. Set-rating behavior is also influenced by some specific items contained in the set. The interwind influence of item combinations varies from user to user, which complicates the problem even more. Considering the complex mutual influences of items in a set, the aggregation should be conducted in a personalized way.

We illustrate the process of SPL in Figure 1. Set ratings are *what we have*, which will be used as a supervision signal in training. In order to optimize the model under the guidance of supervision signals, we need to predict users' ratings towards sets. To achieve this, we require two steps: **item preference prediction** and **personalized aggregation**. Predicted item ratings are aggregated in a personalized way to get set-level predictions. Predicted item ratings are *what we want* eventually, but they can only be updated with indirect supervision. Item preference prediction has been fully explored in recent works, so the key of SPL comes down to personalized aggregation, which is *what we model* in this work. Personalized aggregation aims to model the influence of item interplay for users, which is the main focus of solutions for SPL .

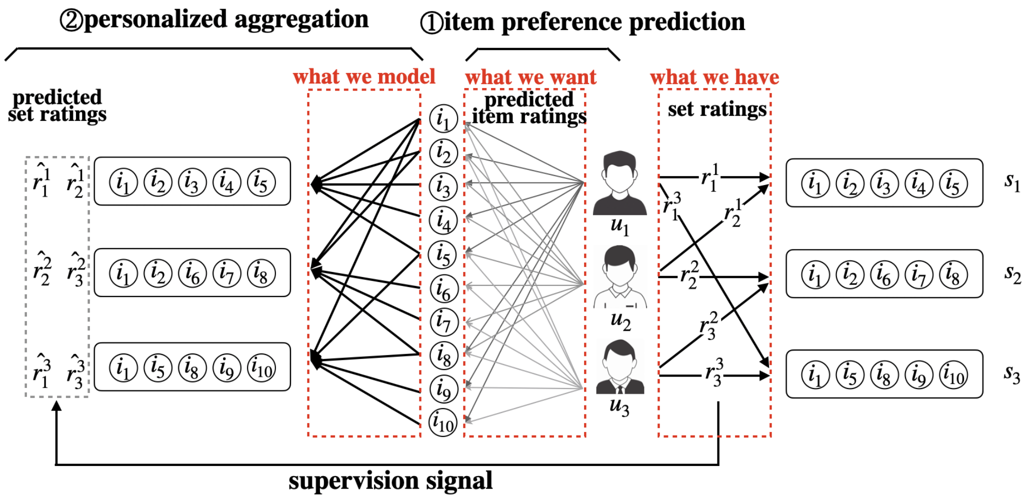
②personalized aggregation
①item preference prediction
what we model
what we want
what we have
predicted set ratings
predicted item ratings
set ratings
$\hat{r}_1^1$ $\hat{r}_2^1$
$\hat{r}_2^2$ $\hat{r}_3^2$
$\hat{r}_1^3$ $\hat{r}_3^3$
i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9 i_{10}
u_1 u_2 u_3
r_1^1 r_2^1 r_1^3 r_2^2 r_3^2 r_3^3
s_1 s_2 s_3
supervision signal

Figure 1. The illustration of *set-based preference learning* (SPL) problem settings. Users $\mathcal{U} = \{u_1, u_2, u_3\}$ rate sets $\mathcal{S} = \{s_1, s_2, s_3\}$, which contain items randomly picked from $\mathcal{I} = \{i_1, i_2, \ldots, i_{10}\}$. The set ratings user u gives, set s, are denoted as r_u^s. $\hat{r}_u^s$ is the *predicted* preference scores for user u's ratings of set s. The prediction process is optimized by minimizing the loss between r_u^s and $\hat{r}_u^s$. $\hat{r}_u^s$ is aggregated from predicted item ratings in a personalized way. As the training process proceeds, the set-rating prediction becomes more accurate. As a by-product, we get more reasonable item-rating predictions. At this point, the goal of inferring user preference over individual items based on set-level feedback is achieved.

Current works addressing the SPL problem focus on designing a personalized aggregation mechanism for item-level preference predictions. Since SPL is a less studied problem, there are only a few heuristic works addressing it, and the main difference between these existing models is how to aggregate the item-level predictions to get set-level predictions. The work in [5] proposed three algorithms of different intuitions. The *average rating model (ARM)* takes an average of all item-level predictions to get set-level predictions. It assumes equal contributions of each item in the set, which ignores the impact of diverse preference of items in the set. The *variance offset average rating model (VOARM)* considers the diversity of item-level predictions by adding a weighted variance term in aggregation. However, it still treats items equally, underemphasizing items that are more influential in set-based preference. Against the shortcomings of ignoring the different item importance in the above methods, the *extremal subset average rating model (ESARM)* enumerates all extremal subsets of item predictions and aggregates these subsets via linear weighted sum. Although considering the different impact of items in set, extremal subset average rating model ignores the fact that the selection of subsets should be highly flexible to reflect personalized assessment strategies. Extremal subsets are not necessarily good choices for all users and items. Moreover, enumerating all extremal subsets causes a huge computation burden as the set size grows. The limitation of the these methods is that they are attempts at one-size-fits-all approaches to aggregate item-level predictions, thereby oversimplifing the interplay between items in aggregation. We argue that the user's rating of a set is largely influenced by subsets arousing the strongest emotional inclinations. For example, some users overrate a playlist because of a few songs that are highly favored; some users underrate the game bundle because of one very disliked game. Moreover, we believe that the influential subsets should be automatically selected in an end-to-end manner, instead of following a predefined rule.

To these ends, we introduce a *self-attentive subset learning model* (SaSLM), which predicts item-level preferences from set-based feedback by capturing the interrelation among items in the set. Due to the computational burden of enumerating all possible combinations, it is nontrivial to find the optimal subsets in a personal way [6–8]. Rather than exhaustion, we propose to explicitly learn a personalized subset selection strategy via a *policy network*. Given the complex correlation between items in the set-rating process, we introduce a self-attention module to make sure all set members are considered in subset selection. Furthermore, we introduce gumble softmax in avoidance of gradient vanishing caused by binary selection in model learning. Then, the selected subsets are aggregated by personalized positional weights, which is a group of parameters specially learned for each user. The weights can be seen as the reflection of a user's rating pattern towards sets.

We summarize the contributions of this paper:

- We propose a novel method, self-attentive subset learning model (SaSLM), to study the problem of set-based preference learning (SPL), which is important to efficiently reveal users' preferences towards items with indirect supervision and protect their privacy as well. To the best of our knowledge, this is the first work to tackle set-based preference learning problem in an end-to-end manner.
- We introduce a policy network to select a representative subset for each item set. By this means, the extent to which users' preferences are influenced by different items is fully considered. Our proposed subset selection strategy is more flexible.
- We propose a novel personalized aggregation method to turn item-level predictions into set-level predictions by user-specific personalized positional weights, which is more efficient and has few parameters.
- Through extensive empirical evaluation, the effectiveness of SaSLM at predicting item-level preferences under set-based preference supervision has been confirmed.

In what follows, we organized the paper according to the roadmap: Section 2 surveys the work that is relevant to the problem studied in this paper. Section 3 introduces the notations and definitions. We first overview our model and then provide the details in Section 4. We describe the experimental settings in Section 5 and report and analyze the experimental results in Section 6. In Section 7, we conclude our paper.

2. Related Work

In this section, we provide a brief survey of the relevant research. We first introduce the current progress in set-based preference learning (SPL), and then draw distinctions between it and a similar topic, bundle recommendation (BR).

2.1. Set-Preference Learning

A few lines of work propose to infer preferences of items from the preferences of item sets [5–9]. The most relevant one is [5], which collects user set-level ratings by sending emails to users. The sets are randomly constructed to a fixed size (five items in this case) from the rated movies in the user's history. Noteworthily, constructing sets from users' previously rated items does not change the setting of SPL. Still, the goal is to reduce item ratings from set ratings, though the item ratings will be used to evaluate the accuracy of item-rating prediction. To infer item-level preferences, the author of [5] proposed three different methods:

* *average rating model* (ARM) [5] assumes that a set-based rating reflects users' average ratings for all the items. The estimated rating of user u for set S is given by the average of all predicted item ratings:

$$\hat{r}_u^s = \mu_u^s = \frac{1}{|S|}\sum_{i \in S} \hat{r}_{ui} \tag{1}$$

* *variance offset average rating model* (VOARM) [5] captures the preference diversity of items in the set. Besides estimating the mean of the set rating, it also estimates the variance. Therefore, the final prediction is given by:

$$\hat{r}_u^s = \mu_u^s + \beta_u \sigma_u^s, \tag{2}$$

where β_u is the personalized factor; μ_u^s is mentioned in Equation (1) as the mean of the ratings of all items, and σ_u^s is the standard deviation of the ratings of items in the set S, which is given by:

$$\sigma_u^s = \sqrt{\frac{1}{|S|}\sum_{i \in S} (\hat{r}_{ui} - \mu_u^s)^2}.$$

* *extremal subset average rating model* (ESARM) [5] estimates the ratings of extremal subsets and predicts set rating with weighted aggregation:

$$\hat{r}_u^s = \sum_{i=1}^{2n_s - 1} w_{ui} e_{ui}, \tag{3}$$

where e_{ui} is the average rating of the i-th extremal subset.

The method proposed by [9] constructs groups of items to efficiently elicit the initial preferences of cold-start users. Then, a new user's interest in the individual items is generated through the average preference of users who share similar tastes for the item groups. Differently from [9], SPL focused on a more general case: (1) rather than constructing sets with similar items, SPL does not require homogenous items, and (2) SPL aims to develop fully personalized recommendation methods that are not limited to cold-start recommendation.

There are also some works in the multi-agent systems community focusing on representing set-level preferences [6–8]. Differently from SPL, these methods aim to find the optimal subsets by partial order over the power set, which requires item-level feedback. In comparison, SPL only assumes the availability of set-level feedback, so the partial order cannot be obtained.

2.2. Bundle Recommendation

bundle recommendation (BR) is a similar topic to SPL, as they both study the set-level feedback, but with different goals. BR aims to learn users' preferences for sets (bundles), so that users can consume the whole set in one go, whereas the goal of SPL is to learn the preferences for individual items from set-level feedback.

Most works leveraging set-level feedback focus on the problem of BR, since the problem has item-level feedback as direct supervision. BR has wide applications in recommender systems, ranging from the recommendations of music playlists [10,11], travel packages [12,13], and item baskets [14] to the recommendation of user-generated lists [15–17], etc. Generally, these methods rely on the interactions between users and item sets and take advantage of set-level collaborative filtering [15,18,19]. Recent works also integrate user–item interactions. By sharing model parameters, multi-task learning methods can be formed [20,21]. Given past sequential sets of items, some methods [14,22] learn to predict the subsequent sets. Applications include the next basket recommendation [23–25], next clinical event prediction [26,27], and next repeat set prediction [28].

3. Preliminaries

In this section, we introduce the notation that we use in this paper, which we summarize in Table 1.

Table 1. Notation.

Notation	Description
$\mathcal{U}$	collection of users
$\mathcal{I}$	collection of items
$\mathfrak{S}$	collection of item sets
$S \in \mathfrak{S}$	one set of items
n_u	number of users, i.e., $n_u = \lvert U \rvert$
n_i	number of items, i.e., $n_u = \lvert I \rvert$
n_s	maximum length of item sets
d	dimension of embeddings
R^s	collection of rating scores s
r_u^s	the rating score that user u provided to set s
$\hat{r}_u^s$	the predicted score of user u to set s
$\hat{r}_u^i$	the predicted score of user u to item i
x_{ui}^s	the variable to indicate whether item i in the set S will be selected in the subset for user u
$\boldsymbol{p}_u \in \mathbb{R}^d$	embedding of user u
$\boldsymbol{q}_i \in \mathbb{R}^d$	embedding of item i
$\boldsymbol{w}_u \in \mathbb{R}^{n_s}$	personalized positional weights of user u
$P \in \mathbb{R}^{n_u \times d}$	embeddings of all users for item-level preference modelling
$Q \in \mathbb{R}^{n_i \times d}$	embeddings of all items for item-level preference modelling
$P' \in \mathbb{R}^{n_u \times d}$	embeddings of all users for subset selection
$Q' \in \mathbb{R}^{n_i \times d}$	embeddings of all items for subset selection
$W \in \mathbb{R}^{n_u \times n_s}$	personalized positional weights

We use $\mathcal{U}$ and $\mathcal{I}$ to denote all users and all items, respectively. We use $\mathfrak{S}$ to denote the collection of item sets. The numbers of users and items are denoted by n_u and n_i, i.e., $n_u = |\mathcal{U}|$ and $n_i = |\mathcal{I}|$. For different sets of items, the number of items can be different. Thus, we use n_s to denote the maximum length of item sets. We use R^s to denote a collection of rating scores and use r_u^s to denote the rating that user u provides to the set $S \in \mathfrak{S}$. To distinguish with the true rating, we use $\hat{r}_u^s$ to represent the predicted rating score. Similarly, the predicted rating score of user u to item i is denoted by $\hat{r}_u^i$. We use x_{ui}^s as an indicator variable that determines whether item i in the set S will be selected in the subset for user u or not.

We denote the dimensions of embeddings by d. We write $\boldsymbol{p}_u \in \mathbb{R}^d$ and $\boldsymbol{q}_i \in \mathbb{R}^d$ as the embeddings for user u and item i, respectively. We pack all the embeddings of users to form the matrix $P \in \mathbb{R}^{n_u \times d}$. Similarly, we can define matrix $Q \in \mathbb{R}^{n_i \times d}$ by packing the embeddings of all items. P and Q are used to predict item-level preferences in the rating prediction module. We also define the matrix $P' \in \mathbb{R}^{n_u \times d}$ and $Q' \in \mathbb{R}^{n_i \times d}$ to select subsets in aggregating item-level ratings for set ratings. Noteworthily, these two groups of user and item embeddings are specially designed for different purposes, namely, rating prediction and subset selection. We also define $\boldsymbol{w}_u \in \mathbb{R}^{n_s}$ as the personalized positional weights for user u. We learn for each user n_s positional weights, one for each position in the set. We can define matrix $W \in \mathbb{R}^{n_u \times n_s}$ by packing all the weights.

We define the problem of set-based preference learning (SPL) formally: *given a collection of set ratings, the goal of SPL is to accurately learn the user's preferences for individual items without the direct supervision of item-level feedback.*

Input. Collection of users $\mathcal{U}$; collection of rating scores R^s.

Output. A personalized scoring function that maps each item to a real value score for each user:

$$\hat{r}_u^i = f(u, i | R^s, \mathcal{U}, \mathfrak{S}, \mathcal{I}) \tag{4}$$

$\hat{r}_u^i$ is the predicted rating score of user u for item i.

4. Model Description

In this section, we present the *self-attentive subset learning model* (SaSLM), as shown by an illustrative overview in Figure 2, which is composed of three components: (1) the **rating prediction module**, which predicts item-level ratings by collaborative filtering (CF); (2) the **subset learning module** that selects a most representative subset from the original set; and (3) the **rating aggregation module** integrates results from (1) and (2) to predict the user's set-level rating.

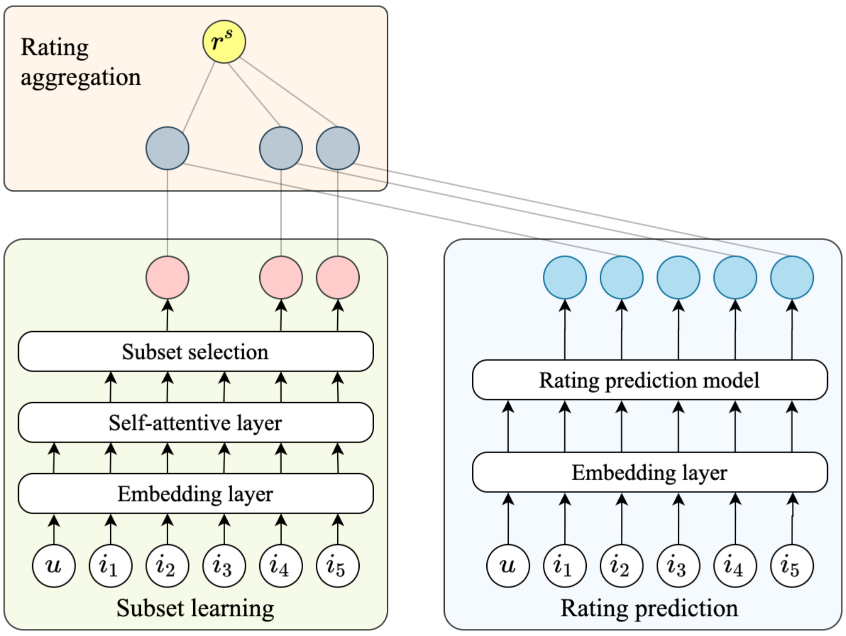

Figure 2. Model overview of the SaSLM, which consists of three components: rating prediction, subset learning, and rating aggregation. **The rating prediction module** predicts item-level ratings by matrix factorizing score function. **The subset learning module** is the key of the proposed model, which outputs a representative subset in an end-to-end manner. **The rating aggregation module** predicts users' set-level ratings according to the item-rating predictions and subset selection results from two other modules. Finally, the model is optimized under the supervision of set ratings that users give.

The goal of SPL is to learn item-level preference by set-based feedback. Therefore, for SPL, the explicit generation of item-level prediction scores is required (*rating prediction module*). However, due to the absence of item-level supervision signals, the predicted scores of individual items should further be aggregated to form the prediction of set-level ratings (*rating aggregation module*). Since the set-level ratings are better revealed by a subset of items in the set [5], to reduce the bias between set-level ratings and item-level ratings, SaSLM learns a subset of items from the set (*subset learning module*).

4.1. Rating Prediction

Item-rating prediction has been widely studied, where state-of-the-art performance has been achieved by CF methods [2,3,29–31]. A general form of rating prediction can be formulated as follows:

$$\hat{r}_{ui} = f(\boldsymbol{p}_u, \boldsymbol{q}_i), \tag{5}$$

where $\boldsymbol{p}_u, \boldsymbol{q}_i \in \mathbb{R}^d$ are the embeddings of user u and item i, respectively. $f(\cdot)$ is the score function. As one of the widely used CF methods, matrix factorization (MF) defines $f(\cdot)$ as the inner product:

$$f(\boldsymbol{p}_u, \boldsymbol{q}_i) = \boldsymbol{p}_u^\top \boldsymbol{q}_i.$$

Here we follow the method of MF to predict item-level ratings, because it is simple and effective. More complex scoring functions can be saved for future exploration.

4.2. Self-Attentive Subset Learning

As mentioned before, to simulate the decision-making process of set rating, it is reasonable to select a subset of items to reduce the bias between set-level and item-level preferences [5]. Then, the set rating will be predicted based on the chosen subset.

To choose subsets, we propose a *self-attentive policy network* (SaPN). SaPN learns the personalized policy for how users select subsets from sets. Figure 3 illustrates the network structure. Note that we borrow the concept of the *policy network* form reinforcement learning (RL), which takes the current state as input and outputs the probability distributions of all possible actions [32]. Intrinsically, rather than generating probabilities, SaPN learns to generate the selected subset directly based on the *Gumbel-softmax* trick [33]. The SaPN consists of three parts: (1) an embedding layer that implicitly models user's preference and item attributes; (2) a self-attention layer that takes all items into consideration while choosing subsets; (3) a policy network layer that samples the chosen subset according to the reparameterization sample.

The SaPN is an important part of SaSLM. This section will provide more details layer by layer.

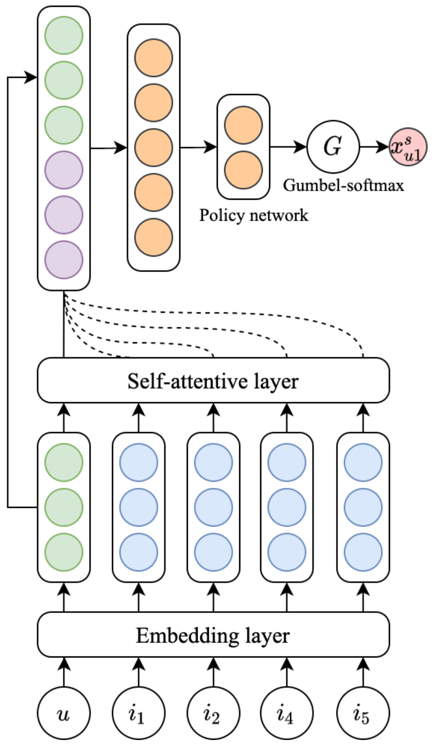

Figure 3. An illustrative example of the self-attentive policy network. It takes as input the user u and items in the set $S = \{i_1, i_2, i_3, i_4, i_5\}$. It shows how the indicator variable x^s_{u1} is generated.

Embedding layer. Given a user u and a set containing several items $\{i_j\}_{j\in S}$, the embedding layer takes the one-hot representations as input. We assume $\boldsymbol{u} \in \mathbb{R}^{n_u \times 1}$ is the one-hot vector of user u, where only the element corresponding to u is one, and other elements are all zero. Similarly, we have the one-hot vectors of items, i.e., $\boldsymbol{i}_j \in \mathbb{R}^{n_i \times 1}$. Through an embedding layer, we can retrieve the embeddings of users and items. The embedding layer maintains two matrices, user matrix $P' \in \mathbb{R}^{n_u \times d}$ and item matrix $Q' \in \mathbb{R}^{n_i \times d}$, respectively. The output of the embedding layer is the embeddings of user u and item i_j:

$$\boldsymbol{p}'_u = P'^{\top}\boldsymbol{u}, \quad \boldsymbol{q}'_j = Q'^{\top}\boldsymbol{i}_j.$$

Note that for rating prediction, we also learn user and item matrices P and Q. One possible way is to share the embeddings. However, to ensure that the learning of subsets is independent from the prediction of item-level ratings, we learn another set of embeddings, which we distinguish as P' and Q'.

Self-attention layer. For each item, whether it is to be selected in the subset is determined in turn. However, when choosing the subset, it is important to take all items in the set into consideration. In order to capture correlations among contained items, we apply self-attention. Thus, by transforming item embeddings through self-attentive layer, when determining whether to select an item, other items in the set can be also considered.

Aside from the correlations of other items, we also hope that user preferences can be considered when choosing the subset. To be more specific, given the same set, its contained item embeddings should be catered for different users. The intuition is that users have different levels of focus on the same set when they rate. Therefore, we first multiply $\boldsymbol{q}'_j$ with $\boldsymbol{p}'_u$ and then pack the item embeddings into a matrix, denoted by $Z_s \in \mathbb{R}^{|S| \times d}$:

$$V_s = [\boldsymbol{p}'_u \circ \boldsymbol{q}'_1, \ldots, \boldsymbol{p}'_u \circ \boldsymbol{q}'_{|S|}]^{\top}.$$

Following [34], we use scaled dot-product attention to calculate the self-attention score:

$$Z_s = \mathrm{softmax}\left(\frac{V_s V_s^{\top}}{\sqrt{d}}\right),$$

where $Z_s \in \mathbb{R}^{|S| \times |S|}$ is a self-attention score matrix which indicates the similarities among $|S|$ items in set S. $\sqrt{d}$ is used to scale the attention score for preventing gradient vanishing. The softmax function is to normalize the self-attention scores. The output of the self-attention layer is the matrix product of Z_s with V_s. To stabilize the training as [35], residual connections [36] are obtained as the final output:

$$V'_s = V_s + Z_s V_s.$$

Policy network. After transforming item embedding through the self-attention layer, we obtain the set-aware item embeddings $\boldsymbol{v}'_s$. To learn the user's personalized subset selection strategy, the item embeddings are concatenated with the user embedding, i.e., $[\boldsymbol{p}'_u; \boldsymbol{v}'_s]$. Then, the concatenated embeddings are fed into the policy network.

Typically, a policy network outputs the probability distribution of all possible actions. In our case, it gives the probability of being chosen in the subset, $p(x_{ui} = 1)$, or not, $p(x_{ui} = 0)$. Then, the policy network selects items by sampling from the probabilities. However, the sampling of items involves discrete variables, so that the gradient is unable to propagate backwards. Another straightforward solution is to rely on a naïve Monte–Carlo gradient estimator [37]. The naïve Monte–Carlo gradient estimator samples the discrete variable x_{uj} from $p(x \mid \boldsymbol{p}_u, \boldsymbol{z}_j)$ multiple times to estimate the gradient. It can be understood as the REINFORCE algorithm [38] with a single decision. However, as pointed out in [39], the naïve Monte–Carlo gradient estimator exhibits very high variance, which destablizes the sampling process.

To reduce the variance, the reparameterization trick is generally utilized [39]. Rather than sampling from the data-dependent distribution $p(x \mid \boldsymbol{p}_u, \boldsymbol{z}_j)$, reparameterization samples a random variable ϵ from the data-independent distribution, say, a standard normal distribution. Then, variable x_{uj} is sampled by a function involving both ϵ and $\boldsymbol{p}_u, \boldsymbol{z}_j$. Since the sampled ϵ is independent and identically distributed, the variance during estimation can be largely reduced. For reparameterizing categorical variables in our case, Gumbel-softmax is often utilized [33]. Given the probabilities $p(x_{uj} = 1) = \pi_{uj}$ and $p(x_{uj} = 0) = 1 - \pi_{uj}$, Gumbel-softmax performs reparameterization in the following way:

$$\begin{aligned} \epsilon &\sim \text{Uniform}(0,1), \\ g &= -\log(-\log(\epsilon)), \\ x_{uj} &= \text{softmax}\big((\log(\pi_{uj}) + g)/\tau\big). \end{aligned}$$

Note that x_{uj} is not exactly binary but close to zero or one. τ, named "temperature", is the hyper-parameter for controlling to what extent x_{uj} is close to binary.

At this point, we get the personalized optimal subset from the self-attentive subset learning module. By doing so, we model the pattern how item-level preferences influence users' set-level feedback. It is the key to address the SPL problem by inferring item-level feedback from set-based preferences.

4.3. Position-Aware Rating Aggregation

When predicting the scores for individual items in Section 4.1, there are no supervised signals on individual items for learning. Instead, we use set ratings as supervision in SPL. Therefore, we need to predict the set-level ratings in order to learn the model. Thus, we present a subset selection strategy in Section 4.2 to reproduce the process in which users rate sets according to their item-level preferences. After predicting item-level scores and deciding on the subset, we discuss the way to aggregate item-level scores in the position-aware rating aggregation module. To do so, the aggregation of items in the set is required.

Item ratings are generated explicitly, since the aggregation is in the granularity of ratings. Given item ratings, the rating aggregation plays the role of estimating set ratings. In theory, various forms can be defined for the estimation, e.g., regression and neural network. However, these high-level aggregation can introduce certain bias between item ratings and set ratings. Since the ultimate goal of SPL is to estimate item-level preferences, the *unbiased estimation* is required. In this paper, we propose to select subset of items to estimate the set rating. To ensure the unbiased estimation as well, we define the $\oplus$ operation as follows:

$$\oplus_{i=1}^{|S|} \hat{r}_{ui} = \frac{1}{\sum_{i=1}^{|S|} x_{ui}^{s}} \sum_{i=1}^{|S|} x_{ui}^{s} \cdot \hat{r}_{ui}. \tag{6}$$

where $\oplus$ is an aggregation operation to combine items in the set. It is straightforward that Equation (6) is unbiased. x_{ui}^{s} is the variable to indicate whether item i is selected in the subset or not.

While we define x_{ui}^{s} to indicate whether item i is selected in the subset or not, the hard selection brings instability for training. To ensure the smoothness during training, one possible way is to learn soft selection, i.e., learning a weight w_{ui} for each item i in the set. However, soft selection deviates from the idea of subset learning. Differently from learning soft assignments, we propose to learn a folded selection variable. We define x_{ui}^{s} in the following way:

$$x_{ui}^{s} = \begin{cases} w_{ui}, & \text{if the item is selected in the subset,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Equation (7) means that x^s_{ui} is a learnable weight parameter as long as item i is selected. x^s_{ui} will be folded if i is not selected.

Learning for each user–item pair (u, i) a separate weight w_{ui} is infeasible due to the large parameter space. Thus, we propose to learn user-specific weights. We further introduce **personalized positional weights** to aggregate ratings of items in the subset—that is, for each position p in the set, we will learn a weight w_{up}. The positions of items are determined by sorting items according to their predicted ratings. Since the weights are not specific to items, the parameter space can be reduced greatly.

Aside from smoothness, learning positional weights also helps to better fit the set ratings, given item ratings. For example, the positional weights corresponding to larger item ratings can be lowered if a user tends to overrate sets. The positional-aware weights require $O(n_u \cdot n_s)$ parameter space rather than $O(n_u \cdot n_i)$ $(n_s \ll n_i)$.

We illustrate our proposed positional-aware rating aggregation in Figure 4. Given the ratings of items in the selected subset, e.g., $\{\hat{r}_{u2}, \hat{r}_{u4}, \hat{r}_{u5}\}$, we first sort the ratings. Through ranking, we can generate the ordered ratings $\hat{r}_{u4}, \hat{r}_{u5}, \hat{r}_{u2}$ if $\hat{r}_{u4} > \hat{r}_{u5} > \hat{r}_{u2}$. Accordingly, we also retrieve three positional weights from $\boldsymbol{w}_u$; i.e., w_{u1}, w_{u2}, w_{u3}. Therefore, we can predict the ratings by user u to set S as:

$$\hat{r}^s_u = \frac{w_{u1}\hat{r}_{u4} + w_{u2}\hat{r}_{u5} + w_{u3}\hat{r}_{u2}}{w_{u1} + w_{u2} + w_{u3}}.$$

These user-specific positional weights are used to aggregate item-level rating predictions (rating prediction module in Section 4.1) according to the subset selection results (self-attentive subset learning module in Section 4.2). Then, the aggregated set ratings can be used to update parameters. With the loss of set ratings diminishing, SaSLM naturally learns more precise item-level ratings and personalized aggregating patterns. Thus, we reach the goal of SPL, inferring item-level preferences by set-based feedback.

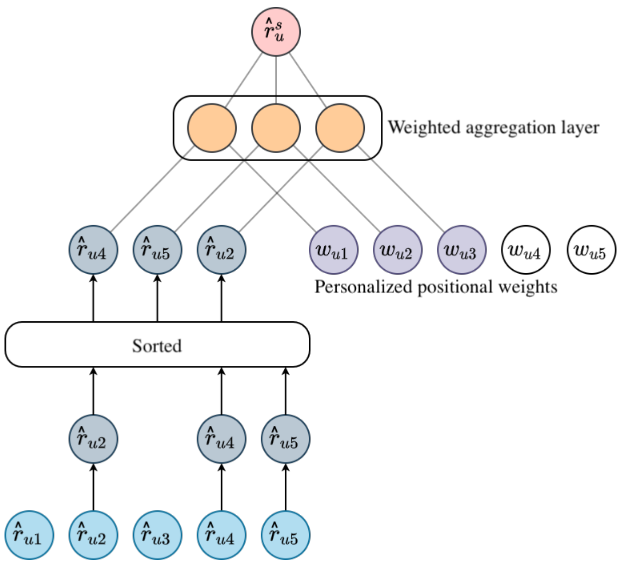

Figure 4. An illustrative example of position-aware rating aggregation. Suppose items i_2, i_4, i_5 are selected from a set of items $\{i_1, i_2, i_3, i_4, i_5\}$ by SaPN. The set-rating predictions $\hat{r}^s_u$ are obtained by aggregating i_2, i_4, i_5 with personalized positional weights w_{u1}, w_{u2}, w_{u3}.

4.4. Model Learning

We define Θ as the parameters associated with the rating prediction model (P, Q and parameters in the score function Equation (5)) and Φ as the parameters in subset selection (P', Q' and parameters in SaPN) and rating aggregation (position-aware weights W). The optimization model of SaSLM is formulated as follows:

$$\arg\min_{\Theta,\Phi} \sum_{u\in\mathcal{U}} \sum_{s\in\mathcal{S}} \mathcal{L}(\hat{r}_u^s(\Theta,\Phi), r_u^s) + \Omega(\{\Theta,\Phi\}), \tag{8}$$

$\mathcal{L}(\cdot,\cdot)$ is the point-wise loss function. Although pair-wise [40] or list-wise [41], loss functions are also utilized in collaborative filtering; the goal of SPL is to reveal individual item preferences from the set preference, rather than ranking items. Moreover, this loss function is also consistent with our experimental evaluation where we evaluate whether the predicted item scores can correctly fit the true rating scores provided by user to the item.

$\Omega(\cdot)$ is the regularization function. While various forms of regularization have been studied, we consider the simple but widely used ℓ_F-norm regularization to regularize user and item embeddings and the positional weights:

$$\Omega(\{\Theta,\Phi\}) = \alpha(\|P\|_F^2 + \|Q\|_F^2) + \beta(\|P'\|_F^2 + \|Q'\|_F^2) + \lambda\|W\|.$$

We propose to optimize Θ, Φ in Equation (8) alternatively. This is because the learning of subsets can be relatively unstable due to the discrete selection. Therefore, we isolate the learning of subsets with the learning of rating prediction. The positional weights W is contained in Φ. Learning W together with the policy network helps to relieve the problem of instability. On the other hand, by fixing W and the parameters of SaPN, the model can focus on learning item ratings.

5. Experimental Setup

5.1. Dataset

We consider three datasets to evaluate the performance of SaSLM on the task of SPL:

- **RealSet**: The dataset was collected from the MovieLens platform by [5]. Users were selected if they were active, since January 2015, and rated at least 25 movies. The set ratings were collected by sending emails to users. The movie sets were created by randomly selecting five movies without replacement from those they had already rated before.
- **NetEase**: This is a dataset collected from the largest music platform in China by [19]. It enables users to create song bundles or thumbs-up any bundles created by others. We use NetEase as the dataset for SPL by treating the music bundle as the item set.
- **YouShu**: This dataset was constructed by [20] from YouShu, a Chinese book review site. Similarly to NetEase, every bundle is a list of users' desired books. We treat the book list as the item set.

The statistics of these datasets are given in Table 2. Note that among the three datasets, only RealSet is specially designed to evaluate SPL. RealSet contains explicit ratings, ranging from 0.5 to 5, for both items and sets. Although explicit ratings can well reveal the user's preferences, they are not always available in real user interactions. Taking more realistic scenarios into consideration, we also transform the RealSet under implicit settings by binarizing the ratings. To be more specific, ratings no less than three are regarded as positive, and otherwise negative. To distinguish, we use **RealSet**-explicit and **RealSet**-implicit to denote the RealSet dataset with explicit and implicit ratings, respectively.

Since the problem of SPL is underexplored, few datasets are available. Therefore, we constructed datasets from bundle recommendation by discarding user–item interactions for training because SPL aims to predict item ratings by set ratings. In RealSet, all item-level ratings are available for evaluation. However, users just interacted with a small faction of items in the bundle in both NetEase and YouShu datasets. These non-interacted items are not necessarily negative ones but possibly unexposed items. Lacking of negative feedback in latter two datasets, we simply followed the missing-at-random assumption by treating all non-interacted items as negative [3]. Considering the partial observation of positive items, to ensure the balance between positive items and negative items, we sampled an equal number of negative items as positive ones.

Table 2. Data statistics.

Name	#Users	#Items	#Sets	#Ratings
RealSet	853	13,012	29,516	29,516
YouShu	8039	32,770	4368	38,977
NetEase	18,528	123,628	22,864	302,303

5.2. Baselines

To evaluate the performance of SaSLM, we compare SaSLM with the following baseline methods:

- *matrix factorization on set rating* (MFSet): This is the personalized baseline compared in the experiments of [5]. It assumes that if a user rates a set, he/she will give all contained items the same ratings. The MFSet is the matrix factorization (MF) method with same set ratings assigned to all items.
- *average rating model* (ARM): The first method introduced by [5], where the set rating is predicted by the average of predicted scores of items in the set. We have introduced ARM in Equation (1).
- *variance offset average rating model* (VOARM): The second method introduced by [5]. Aside from estimating the mean of ratings of items in the set, VOARM also estimates the variance. We have introduced VOARM in Equation (2).
- *extremal subset average rating model* (ESARM): The third method introduced by [5], which learns the ratings of extremal subsets and aggregates the scores to predict set rating. We have introduced ESARM in Equation (3).
- *bundle collaborative filtering* (BCF): The simple CF method that considers user-bundle interactions for bundle recommendation. BCF aggregates item embeddings to represent the bundle by simply adding all the embeddings. Therefore, the rating of user u on set s is given by:
$$\hat{r}_u^s = \boldsymbol{p}_u^\top \sum_{i \in s} \boldsymbol{q}_i.$$
- *bundle graph convolutional network* (BGCN): The state-of-the-art method for bundle recommendation [21], which relies on graph convolutional network (GCN) [42] for embedding aggregation. Note that for SPL, we do not use user–item interactions for training. Therefore, for BGCN we only construct a user-bundle graph for embedding propagation.

5.3. Evaluation

To evaluate SPL, we follow the method of [5] to formulate it into a rating prediction problem. Given the set rating, SaSLM predicts ratings of all items in the set. Naturally, we consider root mean square error (RMSE) as the metric to evaluate rating prediction. A lower RMSE indicates a better performance. RMSE is defined as the root of mean square error (MSE):

$$\begin{aligned}\text{MSE} &= \frac{1}{|\mathcal{U}| \cdot |S|} \sum_{u \in \mathcal{U}} \sum_{s \in \mathcal{S}} (r_u^s - \hat{r}_u^s)^2 \\ \text{RMSE} &= \sqrt{\text{MSE}}.\end{aligned}$$

We performed a paired sample *t*-test on RMSE to show if the the improvement is significant.

6. Experimental Results

In this section, we seek to answer the following research questions:

RQ1 Does the subset assumption—users' feedback for sets is affected by subsets of items, not the whole set—hold?

RQ2 What is the overall performance of SaSLM for the task of SPL? How does SaSLM perform compared with existing SPL methods?

RQ3 How do different learning modules in SaSLM affect the performance of SPL?

RQ4 How does the pre-training of SaSLM affect the performance of SPL?

6.1. Feasibility of Subset Assumption

In order to answer **RQ1**, we first evaluate the subset learning and rating aggregation in SaSLM to show how well the subset assumption models the set-level interaction. Specifically, we generate the set ratings by using subsets learned from the true item ratings. We used the same data split as described in Section 5.3. Rather than predicting item ratings, we used the true ratings of items by users to predict set ratings. This was for verifying the intuition of subset assumption. Note that using item ratings here is not contradictory to SPL settings, where item ratings are predicted by set ratings. This experiment aimed to verify the fact that when users are mainly affected by some salient items when they rate a set, no matter whether they like these items or not. These items, which are more influential in users' set-rating process, consist of what we call "subsets". By using the item ratings to simulate the set-rating process, we prove that it is necessary to learn personalized subset selection and aggregation for rebuilding users' set-rating process. For a comparison, this experiment was also performed on ARM, VOARM, and ESARM, with different rating aggregations. Results are reported in Table 3. As shown in the table, SaSLM performed best on both explicit and implicit RealSet. In comparison with ARM and VOARM, which do not learn subsets, the subset learning in SaSLM can well reflect users' underlining patterns of set-rating behaviors. Compared with ESARM, which learns extremal subsets, the subset learning in SaSLM is more effective, since it learns personalized subsets for users. The results reported in Table 3 also show the performance ceilings of the methods for set-rating prediction.

Table 3. Fitting set ratings using different methods.

Method	RealSet-Explicit	RealSet-Implicit
ARM	0.5861	0.4042
VOARM	0.5250	0.3860
ESARM	0.5176	0.3742
SaSLM	**0.5151**	**0.3712**

6.2. Overall Performance of SaSLM

In this section, in order to answer **RQ2**, we evaluate SaSLM for SPL, where only set-level ratings are available. We report the predictions of both set-level ratings and item-level ratings. Since we learned from set-level ratings, the set-level performance shows the learning capabilities of the different models. In comparison, the item-level results show the performance in our proposed SPL problem. As we have discussed in Section 4.1, we tried both MF and neural collaborative filtering (NCF) for rating prediction. Therefore, we append the postfix -MF or -NCF to distinguish the SPL-specific methods. We also compare these with two baseline methods, MFSet and BCF.

6.2.1. Performance with RealSet

Note that we do not compare our method with BGCN for this experiment. The reason is that BGCN was originally designed for set-based recommendation, which requires constructing the user-bundle graph. However, the graph cannot be constructed for RealSet, since every set is randomly constructed, and therefore each user will interact with a unique set.

Results are recorded in Table 4. We first look at the comparison on RealSet-explicit. As shown in the table, although SaSLM could not outperform the baselines for set-rating prediction, it always showed significant improvement over the compared methods for item-rating prediction. Since the ultimate goal of SPL is to reveal users' preferences for individual items, SaSLM shows its effectiveness for this task. Generally, SPL methods with MF had good performance, except that the second-lowest RMSE for set-rating prediction was achieved by BCF. While this can be counter-intuitive, since the methods with the more expressive NCF had poorer performance, it is justifiable for SPL. The reason is that the expressiveness of NCF can be shown only if enough strong supervision can be provided. However, RealSet contains relatively sparse data to train neural methods and SPL contains no direct supervision to reveal item-level preference.

Table 4. Performance of set-rating prediction with fixed-size sets

	Method	**Set-Level**		**Item-Level**	
		RMSE	***p*-Value**	**RMSE**	***p*-Value**
RealSet-Explicit	MFSet	0.6340	–	0.9369	–
	BCF	$\underline{0.6285}$	–	0.9296	–
	ARM-MF	0.6294	0.2633	0.9274 ▲	0.0
	VOARM-MF	0.6333	0.4071	0.9152 ▲	8.3×10^{-69}
	ESARM-MF	**0.6268**	0.2230	0.9240 ▲	6.6×10^{-54}
	SaSLM-MF	0.6316	–	**0.9103**	–
	ARM-NCF	0.6326	0.6439	0.9427 ▲	0.0
	VOARM-NCF	0.6343	0.5771	0.9325 ▲	0.0
	ESARM-NCF	0.6340	0.4569	0.9246 ▲	0.0004
	SaSLM-NCF	0.6336	–	$\underline{0.9233}$	–
RealSet-Implicit	MFSet	$\underline{0.3967}$	–	0.4841	–
	BCF	0.3970	–	0.4795	–
	ARM-MF	0.3973 ▼	9.8×10^{-6}	0.4810 ▲	0.0
	VOARM-MF	**0.3964** ▼	1.8×10^{-32}	0.4750 ▲	0.0
	ESARM-MF	0.3981 ▼	4.5×10^{-26}	0.4784 ▲	0.0
	SaSLM-MF	0.4001	–	**0.4697**	–
	ARM-NCF	0.4144 ▲	2.9×10^{-14}	0.5111 ▲	0.0
	VOARM-NCF	0.4090 ▲	4.9×10^{-15}	0.4934 ▲	6.6×10^{-40}
	ESARM-NCF	0.4058 ▲	3.4×10^{-12}	0.4793 ▲	0.0
	SaSLM-NCF	0.3977	–	$\underline{0.4702}$	–

We show the best and second best results in **boldface** and $\underline{\text{underlined}}$ text. Statistical significance of pairwise differences in ARM, VOARM, and ESARM over SaSLM were determined by a *t*-test. ▲ indicates the RMSE is significantly higher than that of SaSLM, and ▼ indicates the the RMSE is significantly lower than that of SaSLM. If the *p*-value is lower than 0.01, it is considered significant. The *p*-value is also reported in the table to show the degree of significance. Note that the significant test was conducted using MF or NCF, respectively.

We then analyze the results on the RealSet-implicit dataset. The results are similar to those of RealSet-explicit. SaSLM outperformed other methods significantly for item-rating prediction. The difference is that the lowest and second-lowest RMSE for set-rating prediction were achieved by VOARM and MFSet, respectively. In set-rating prediction, SaSLM significantly outperformed ARM, VOARM and ESARM when using NCF, though being significantly outperformed when using MF. While similar to RealSet-explicit, for which methods with MF generally perform better than those with NCF, SaSLM-NCF achieved a lower RMSE in set-rating prediction than SaSLM-MF. In fact, by sampling to select subsets from sets of items, SaSLM can take advantage of a method similar to dropout to avoid overfitting.

6.2.2. Performance for Bundle Sets

We further ran experiments on item bundles. We analyzed the experimental results on YouShu and NetEase. Since the two datasets were originally constructed for BR, there are also some differences worth mentioning: (1) both Youshu and NetEase contain implicit feedback only; (2) the sets in both datasets contain variable sizes of items; (3) the sets in both datasets were not randomly constructed. Instead, they were constructed for a certain purpose: interaction with multiple users.

Based on these difference, we added BGCN and removed ESARM to/from the comparison. Although, as mentioned in [5], ESARM can be extended to variable-length sets, it has certain implementation ambiguity. ESARM learns personalized weights for each extremal subset. However, for sets with variable length, the number of extremal subsets is undetermined. Therefore, it is not clear how to align the weights to the extremal scores. On the other hand, since the sets in YouShu and NetEase relate to multiple users, the user-bundle graph can be constructed. Therefore, we implemented BGCN by propagating over the user-bundle graph.

We report the results in Table 5. The best performance in item-rating prediction of both datasets was also achieved by SaSLM. Compared with RealSet, the baseline methods had better results in set-rating prediction, possibly due to the collaborative filtering among user–set interactions. Similarly, VOARM generally fit the set ratings well, since it considers the item rating diversity during rating aggregation. ARM and SaSLM had similar performances, both with MF and with NCF. While their performances in set-rating prediction were the same, SaSLM performed slightly better than ARM in item-rating prediction.

Table 5. Performance of set-rating prediction with sets of various sizes.

	Method	**Set-Level RMSE**	**Item-Level RMSE**
YouShu	MFSet	0.4690	0.5680
	BCF	<u>0.4216</u>	0.7045
	BGCN	0.4467	0.7082
	ARM-MF	0.4979	<u>0.5013</u>
	VOARM-MF	0.5000	0.5018
	SaSLM-MF	0.4979	**0.5000**
	ARM-NCF	0.4343	0.6083
	VOARM-NCF	**0.4184**	0.7794
	SaSLM-NCF	0.5586	0.5133
NetEase	MFSet	0.4881	0.5128
	BCF	**0.4074**	0.7035
	BGCN	0.6305	0.7073
	ARM-MF	0.4985	<u>0.5002</u>
	VOARM-MF	0.4953	0.5005
	SaSLM-MF	0.4985	**0.5001**
	ARM-NCF	0.4517	0.6535
	VOARM-NCF	<u>0.4456</u>	0.7200
	SaSLM-NCF	0.4461	0.6058

6.3. Ablation Study

To further investigate the effectiveness of SaSLM and answer **RQ3**, we conducted an ablation study. We investigated the effects of personalized positional weights and the self-attentive layer. The positional weights account for the order of the predicted item ratings, and the self-attention ensures the awareness of other items in the set during subset selection. We first define subset learning model (SLM), which is the basic model of SaSLM without positional-weights and self-attention. We added up positional-weights and self-attention to SLM, denoted by SLM+*pos* and SLM+*att*, respectively.

Results are reported in Table 6, where performances in both set rating predication and item-rating prediction are shown. SaSLM generally performed the best, except on the NetEase dataset.

As shown in Table 6, results on both RealSet-explicit and RealSet-implicit datasets are similar. Without positional weights and self-attention, SLM performs poorly. Only adding self-attention to SLM did not improve the performance. In comparison with positional weights, the performance improved greatly. By combining positional weights and self-attention, the best results were achieved by SaSLM.

On the YouShu dataset, however, the positional weights deteriorated the results. Although SaSLM performed the best, its advancement mostly came from the self-attentive layer. Similarly, on the NetEase dataset, the positional weights did not provide further performance gain, whereas applying self-attention alone greatly improved the performance and achieved the best results.

In short, both the positional weights and the self-attentive layer play important roles in SPL. The positional weights were effective on RealSet but not on YouShu and NetEase, indicating that the positional weights should be considered only when the sets contain a few items. Unlike positional weights, the self-attentive layer shows effectiveness when the set contains more items.

Table 6. Ablation study of SaSLM.

	Method	**Set-Level RMSE**	**Item-Level RMSE**
RealSet- explicit	SLM	0.6583	0.9181
	SLM+*pos*	<u>0.6342</u>	<u>0.9160</u>
	SLM+*att*	0.6588	0.9188
	SaSLM	**0.6316**	**0.9103**
RealSet- implicit	SLM	0.4181	0.4701
	SLM+*pos*	<u>0.4003</u>	<u>0.4698</u>
	SLM+*att*	0.4180	0.4702
	SaSLM	**0.4001**	**0.4697**
YouShu	SLM	0.5455	0.5333
	SLM+*pos*	0.5669	0.5007
	SLM+*att*	<u>0.5025</u>	**0.4887**
	SaSLM	**0.4979**	<u>0.5000</u>
NetEase	SLM	0.5879	0.5001
	SLM+*pos*	0.5879	0.5001
	SLM+*att*	**0.4985**	0.5001
	SaSLM	0.5876	0.5001

6.4. Impact of Pre-Training

The above experiments show that the pre-training of SaSLM plays an important role in improving final performance. Therefore, we investigated the impact of pre-training in detail to answer **RQ4**. Specifically, we pretrained the rating prediction module of SaSLM via ARM, VOARM, and ESARM. For comparison, we also train SaSLM without pre-training.

The statistics in Figures 5 and 6 show that the performance of SaSLM can be improved greatly with pre-training. This is partially because pre-training embeddings decrease the possible instability of the process of learning subsets. Although the two modules, subset learning and rating prediction, are optimized alternatively, the subset learning module is primarily optimized based on the item ratings generated by the randomly initialized rating prediction model. By pre-training with different methods, the results achieved by SaSLM were also slightly different.

We first analyze the performance in set-rating prediction, as shown in Figure 5. On the RealSet-explicit dataset (Figure 5a,b), pre-training with ESARM led to the best performance. However, on the RealSet-implicit dataset (Figure 5c,d), the situation was the opposite: pre-training by ESARM resulted in worse performance than the other pre-training methods. This is justifiable, since, as shown in Section 6.1, ESARM generates the best set-rating prediction for RealSet-explicit but is outperformed by ARM and VOARM for RealSet-implicit. This shows that the initial performance achieved by the pre-training method has a close connection with the final performance of SaSLM.

Now, we analyze the results in Figure 6. Pre-training is also important to reveal the item-level preferences. In comparison with set-rating prediction, pre-training by VOARM achieved the best performance in all cases for item-rating prediction. The effect of pre-training by VOARM was well demonstrated when predicting ratings by NCF on the RealSet-implicit dataset (Figure 6d). While pre-training by ARM or ESARM improved the performance marginally, the effect of pre-training by VOARM was evident. Note that the initial performance achieved by VOARM was not necessarily always better than that achieved by ARM or ESARM. A possible reason for pre-training by VOARM being effective is that it considers the diversity of item ratings to avoid the over-training of rating prediction models.

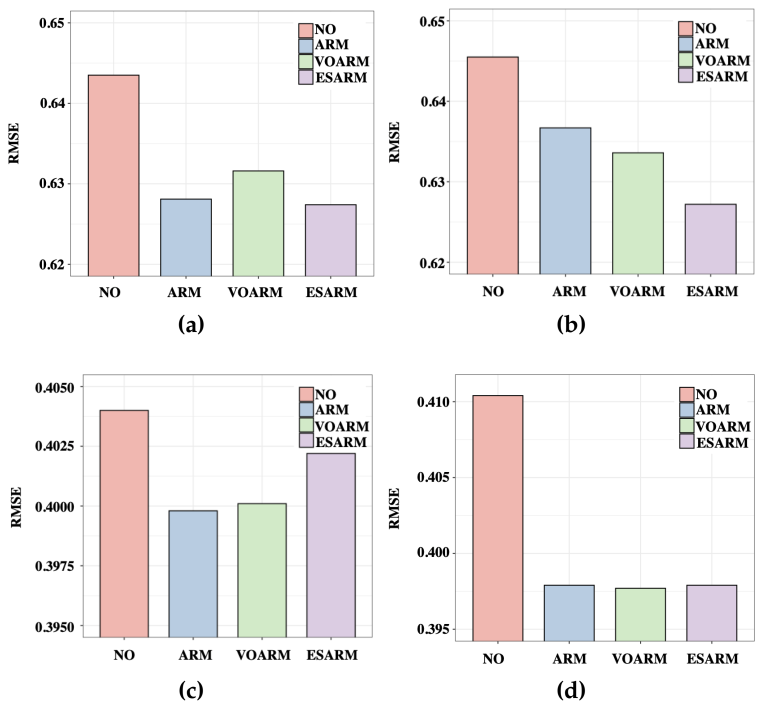

Figure 5. Performance in set-rating prediction under different settings, using different pre-training methods. (**a**) MF, RealSet-explicit. (**b**) NCF, RealSet-explicit. (**c**) MF, RealSet-implicit. (**d**) NCF, RealSet-implicit.

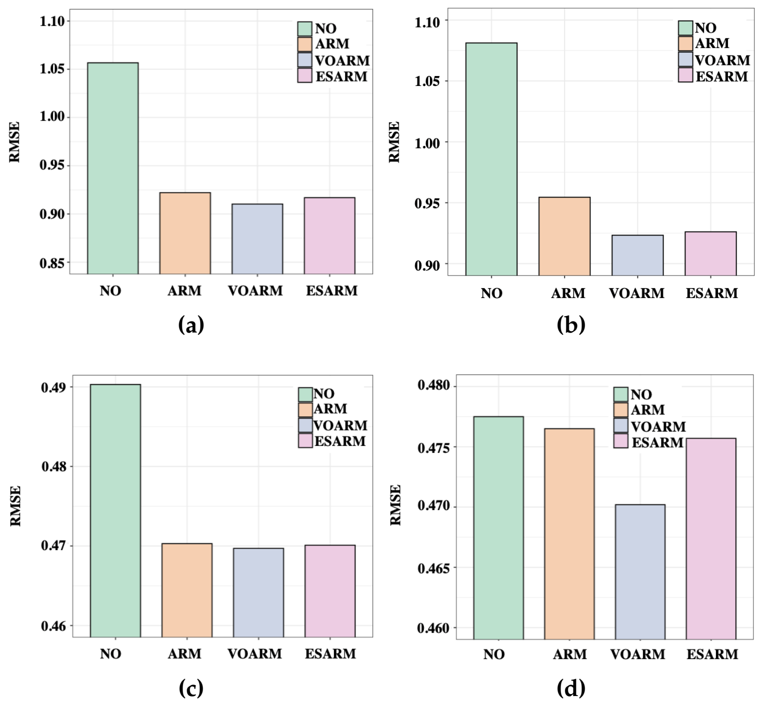

Figure 6. Performance in item-rating prediction under different settings, using different pre-training methods. (**a**) MF, RealSet-explicit. (**b**) NCF, RealSet-explicit. (**c**) MF, RealSet-implicit. (**d**) NCF, RealSet-implicit.

7. Conclusions

In this paper, we studied the problem of set-based preference learning (SPL) for privacy-preserving recommendation, which is important but has been studied little. Observing that the user's preference for the set is usually leveraged by a few items but not the whole set, we proposed the intuition of subset learning to address the problem of SPL. Our solution is the *self-attentive subset learning model* (SaSLM), which consists of a *self-attentive policy network* (SaPN) that learns user subset selection policy. To stabilize the training and ensure the smoothness when fitting set ratings, we also use personalized positional weights to achieve weighted rating aggregation. Unlike existing SPL methods, SaSLM learns personalized and flexible subsets, which largely reduces the gap between set ratings and item ratings. Experiments on real-world datasets demonstrated the effectiveness of SaSLM in the task of SPL.

Since SaSLM mainly focuses on the users' item-level preference disaggregation in a privacy-preserving way, we only evaluated the preference prediction of items contained in sets. In practice, it is important that the item-level preferences can be further propagated to items out of sets. Instead of zero direct item-level supervision, it is also interesting to study the problem when few item-level ratings are available. In future, we plan to perform further research to generalize the definition of SPL and extend SaSLM accordingly.

Author Contributions: Conceptualization, methodology, formal analysis, writing—original draft preparation K.L.; investigation, writing—review and editing, Y.C.; supervision, L.L.; project administration and funding acquisition, J.T. and H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (2020AAA0108800), the National Natural Science Foundation of China (71971212), and the Postgraduate Scientific Research Innovation Project of Hunan Province (CX20200067).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset in this study can be fetched from https://grouplens.org/datasets/learning-from-sets-of-items-2019/, https://music.163.com, https://www.yousuu.com/, all accessed on 28 December 2022.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Aggarwal, C.C. *Recommender Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016.
2. Koren, Y.; Bell, R.M.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *IEEE Comput.* **2009**, *42*, 30–37.
3. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative Filtering for Implicit Feedback Datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 263–272.
4. Himeur, Y.; Sohail, S.S.; Bensaali, F.; Amira, A.; Alazab, M. Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives. *Comput. Secur.* **2022**, *118*, 102746.
5. Sharma, M.; Harper, F.M.; Karypis, G. Learning from Sets of Items in Recommender Systems. *Ksii Trans. Internet Inf. Syst.* **2019**, *9*, 19.
6. desJardins, M.; Eaton, E.; Wagstaff, K. Learning user preferences for sets of objects. In Proceedings of the Machine Learning, Twenty-Third International Conference (ICML 2006), Pittsburgh, PA, USA, 25–29 June 2006; pp. 273–280.
7. Brafman, R.I.; Domshlak, C.; Shimony, S.E.; Silver, Y. Preferences over Sets. In Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, Boston, MA, USA, 16–20 July 2006; pp. 1101–1106.
8. Brewka, G.; Truszczynski, M.; Woltran, S. Representing Preferences Among Sets. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, GA, USA, 11–15 July 2010.
9. Chang, S.; Harper, F.M.; Terveen, L. Using Groups of Items for Preference Elicitation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '15*; Association for Computing Machinery: New York, NY, USA, 2015; pp. 1258–1269.
10. Aizenberg, N.; Koren, Y.; Somekh, O. Build your own music recommender by modeling internet radio streams. In Proceedings of the 21st International Conference on World Wide Web 2012, Lyon, France, 16–20 April 2012; pp. 1–10.
11. Chen, S.; Moore, J.L.; Turnbull, D.; Joachims, T. Playlist prediction via metric embedding. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 714–722.
12. Liu, Q.; Chen, E.; Xiong, H.; Ge, Y.; Li, Z.; Wu, X. A Cocktail Approach for Travel Package Recommendation. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 278–293.
13. Liu, Q.; Ge, Y.; Li, Z.; Chen, E.; Xiong, H. Personalized Travel Package Recommendation. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, Vancouver, BC, Canada, 11–14 December 2011; pp. 407–416.
14. Hu, H.; He, X. Sets2Sets: Learning from Sequential Sets with Neural Networks. In Proceedings of the KDD 2019, Anchorage, AK, USA, 4–8 August 2019; pp. 1491–1499.
15. Liu, Y.; Xie, M.; Lakshmanan, L.V.S. Recommending user generated item lists. In Proceedings of the 8th ACM Conference on Recommender Systems, Silicon Valley, CA, USA, 6–10 October 2014; pp. 185–192.
16. Eksombatchai, C.; Jindal, P.; Liu, J.Z.; Liu, Y.; Sharma, R.; Sugnet, C.; Ulrich, M.; Leskovec, J. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, 23–27 April 2018; pp. 1775–1784.
17. Greene, D.; Reid, F.; Sheridan, G.; Cunningham, P. Supporting the Curation of Twitter User Lists. *arXiv* **2011**, arXiv:1110.1349.
18. Pathak, A.; Gupta, K.; McAuley, J.J. Generating and Personalizing Bundle Recommendations on *Steam*. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, 7–11 August 2017; Kando, N., Sakai, T., Joho, H., Li, H., de Vries, A.P., White, R.W., Eds.; ACM: New York, NY, USA, 2017; pp. 1073–1076.
19. Cao, D.; Nie, L.; He, X.; Wei, X.; Zhu, S.; Chua, T. Embedding Factorization Models for Jointly Recommending Items and User Generated Lists. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 585–594.
20. Chen, L.; Liu, Y.; He, X.; Gao, L.; Zheng, Z. Matching User with Item Set: Collaborative Bundle Recommendation with Deep Attention Network. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019; pp. 2095–2101.
21. Chang, J.; Gao, C.; He, X.; Li, Y.; Jin, D. Bundle Recommendation with Graph Convolutional Networks. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, 25–30 July 2020.
22. Yu, L.; Sun, L.; Du, B.; Liu, C.; Xiong, H.; Lv, W. Predicting Temporal Sets with Deep Neural Networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, 6–10 July 2020.

23. Yu, F.; Liu, Q.; Wu, S.; Wang, L.; Tan, T. A Dynamic Recurrent Model for Next Basket Recommendation. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, 17–21 July 2016; pp. 729–732.
24. Wang, P.; Guo, J.; Lan, Y.; Xu, J.; Wan, S.; Cheng, X. Learning Hierarchical Representation Model for NextBasket Recommendation. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 403–412.
25. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
26. Bai, T.; Zhang, S.; Egleston, B.L.; Vucetic, S. Interpretable Representation Learning for Healthcare via Capturing Disease Progression through Time. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 August 2018; pp. 43–51.
27. Choi, E.; Bahadori, M.T.; Schuetz, A.; Stewart, W.F.; Sun, J. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. In Proceedings of the 1st Machine Learning in Health Care, MLHC 2016, Los Angeles, CA, USA, 19–20 August 2016; pp. 301–318.
28. Benson, A.R.; Kumar, R.; Tomkins, A. Sequences of Sets. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 August 2018; pp. 1148–1157.
29. Sarwar, B.M.; Karypis, G.; Konstan, J.A.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, 1–5 May 2001; pp. 285–295.
30. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, 3–7 April 2017; pp. 173–182.
31. Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434.
32. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
33. Jang, E.; Gu, S.; Poole, B. Categorical Reparameterization with Gumbel-Softmax. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
34. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
35. He, Y.; Wang, J.; Niu, W.; Caverlee, J. A Hierarchical Self-Attentive Model for Recommending User-Generated Item Lists. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, 3–7 November 2019; pp. 1481–1490.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '16, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
37. Paisley, J.W.; Blei, D.M.; Jordan, M.I. Variational Bayesian Inference with Stochastic Search. In Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, 26 June–1 July 2012.
38. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229–256.
39. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the 2nd International Conference on Learning Representations, ICLR '14, Banff, AB, Canada, 14–16 April 2014.
40. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the UAI 2009, the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
41. Yuan, F.; Guo, G.; Jose, J.M.; Chen, L.; Yu, H.; Zhang, W. BoostFM: Boosted Factorization Machines for Top-N Feature-based Recommendation. In Proceedings of the 22nd International Conference on Intelligent User Interfaces, IUI 2017, Limassol, Cyprus, 13–16 March 2017; pp. 45–54.
42. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.