Learning User Embeddings Based on Long Short-Term User Group Modeling for Next-Item Recommendation

Nengjun Zhu[1], Jieyun Huang[1], Jian Cao[2(✉)], and Shanshan Feng[3]

[1] School of Computer Engineering and Science, Shanghai University, 99 Shangda Road, Shanghai 200444, China
{zhu_nj,huang0615}@shu.edu.cn

[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
cao-jian@sjtu.edu.cn

[3] School of Information Science and Engineering, Shandong Normal University, No.1 University Road, Ji'nan 250358, China

Abstract. Session-based recommender systems are increasingly applied to next-item recommendations. However, existing approaches encode session information of each user independently and do not consider the interrelationship between users. This work is based on the intuition that the dynamic groups of like-minded users exist through the time, and users in the same group might share a similar preference. By considering the impact of latent user groups, we can learn a user's preference in a better way. To this end, we propose a recommendation model based on learning user embeddings by modeling long and short-term dynamic latent user groups. It not only captures the latent group information of users, but also perceives the change of user grouping with time. Specifically, we utilize two network units to learn users' long and short-term sessions, respectively. Meanwhile, we employ two additional units to detect which latent groups a user belongs to, followed by an aggregation of these latent group representations. Finally, user preference representations are shaped comprehensively by considering all these four aspects, based on an attention mechanism. Extensive experiments prove our model outperforms multiple state-of-the-art methods in terms of Recall, mAP, and AUC metrics.

Keywords: Session-based recommender · User group modeling · Attention mechanism

1 Introduction

Session-based recommender systems (SBRSs) have been a hot research topic since they can not only model users' long-term preferences, but also highlight

Y. Sun et al. (Eds.): ChineseCSCW 2022, CCIS 1682, pp. 18–32, 2023.
https://doi.org/10.1007/978-981-99-2385-4_2

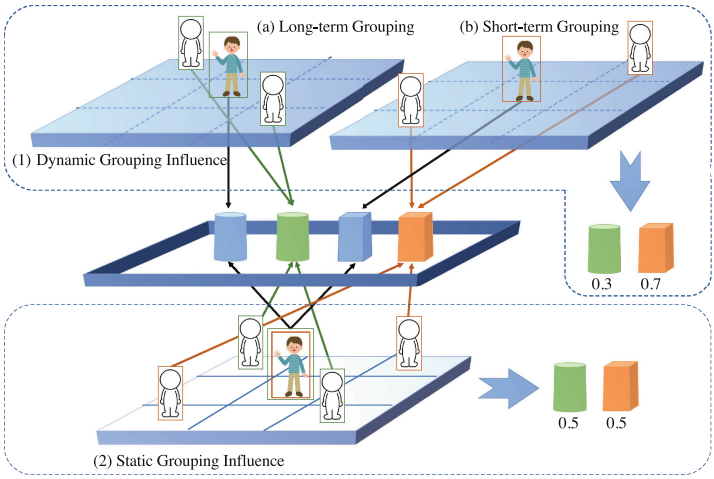


Fig. 1. An example of two different approaches of modeling influences of grouping: (1) our dynamic grouping mechanism, and (2) conventional static grouping mechanism. The central user belongs to two latent groups represented by orange and green frames. In (2), groups influence the user equally and thus the visited items denoted by orange cuboid and green cylinder should be recommended with no difference. In (1), the two groups are treated differently since the central user has switched his group from the green one to the orange one. Up-to-date group might have a larger impact on him and thus items represented by orange cuboid are more in line with his taste. (Color figure online)

short-term demands [1–3]. A session in SBRSs specifies a scope of encapsulation of items, such as a set of products in a shopping cart, a set of viewed websites within a time window, and so forth. Different sessions reflect users' diverse preferences and requirements because users' interests keep changing in various periods [4]. Conventional approaches such as [5,6] treat all sessions equally and overlook the heterogeneity between different sessions, which degrades the performance of next-item recommendations. On the contrary, some SBRSs such as [7–9] distinguish the contribution of each session to depict users' current interests. These systems have demonstrated a decent improvement of recommendation performances compared to conventional approaches.

Most existing SBRSs assume a user's current preference is associated differently with long and short-term sessions. For instance, to learn more complete representations of users, SHAN [9] adopts a hierarchical structure to fuse the pooling results of long and short-term sessions. The fusion weights are differentiated based on an attention network. Similarly, KA-MemNN [7,10] encodes each session from two perspectives: users' intentions and preferences, followed by a more precisely weighted combination of long and short-term session representations. Besides, AttRec [8] and PLASTIC [3] exploit two different prototypes to learn user preferences based on long-term sessions (e.g., using matrix

factorization (MF) [11]) and short-term sessions (e.g., using recurrent neural network (RNN) [12]), respectively.

In all these approaches, user representations are summarized based on their sessions independently, causing the learned models are built on a per user basis. There is no explicit information sharing between the models of users. However, in real-world applications, the groups of like-minded users exist in different contexts. The users in the same group usually share similar preferences and thus might behave similarly. Unfortunately, group information is often neglected in existing SBRSs.

If the data of all related items and users is treated indiscriminately, it is a global model. Instead, more emphasis can be put on some of the related items or users to make the model more targeted, and it is a local model. For example, in [8,13], to capture users' more specific preferences, user representations are learned from currently visited items. At the same time, many local non-session-aware recommender systems (NSRSs) have been widely explored. For instance, based on truncated SVD (singular value decomposition), the work in [14] learns a global model for a shared aspect set as well as a set of user subset specific models. CMN [5] takes users' neighbors as the values in memory network banks, and the values are further accumulated to model users' preferences. Although by considering the stable local influences NSRSs can improve recommendation performance, they still fail to capture users' dynamic preferences and evolving latent groups, and thus are not effective for next-item recommendations. Figure 1 further exhibits the difference between local NSRSs and local SBRSs.

To this end, in this paper, we aim at proposing a local SBRS. However, due to users' complicated behavior patterns, local SBRSs face several challenges: (1) Instead of being assigned a static group like conventional local approaches, in practice, a user might belong to multiple groups. For example, a user can be a cartoon fan and a tech fan at the same time. In such a case, when he purchases a comic book, the taste of cartoon fans would have a more significant impact on him compared to that of tech fans, and vice versa. (2) The interest of each latent user group can be updated over time. For example, a hot cake can disturb the widespread tendency inside the group. (3) Users can switch their latent groups due to multiple reasons, such as the evolution of preferences and requirements. How to capture the dynamics of each latent user group and evolving grouping is not trivial.

To address the problems above, we design a next-item recommendation system (RS) based on modeling long and short-term latent user groups named LSUG. Specifically, we employ a hierarchical neural network to build an end-end representation learning mechanism. We first split the sessions into long and short-term sessions, then we embed each item in sessions into a dense representation. Based on item embeddings, we abstract critical information to form long and short-term session representations by a pooling layer. These representations reflect users' preferences and requirements. Through analyzing the relations between them and latent user group embeddings, we can assign the target user to multiple user groups with different probabilities. Groups with

higher probability have a greater impact on user preferences. Considering that users' interests may be updated over time, as well as their long and short-term preferences, the probabilities of users in groups are dynamic. Then, we can summarize the influences from different user groups by a weighted combination of group embeddings. Finally, the long and short-term session representations and the user group influences are further aggregated to more comprehensive user representations based on an attention model. These representations replace user latent vectors in a pairwise model, i.e., BPR [15], to estimate the probability of an item to be the next visited one. The experimental results show the superiority of our model over multiple state-of-the-art approaches in terms of Recall, AUC, and mAP metrics.

2 Related Work

Traditional approaches, e.g., collaborative filtering (CF), model the relations between users and items in a static way [16]. They neglect the sequential dependencies inside the user-item interactions. To tackle such a problem, Markov chain-based (MC-based) methods such as [17] are developed. The work [18] incorporates hidden Markov models into matrix factorization to deal with temporal dynamics in recommender systems. However, MC-based models only consider the first-order dependency, i.e., it predicts the transition between a pair of items instead of that between an item and a contextual item set. Thus they don't fit the next item recommendation well.

Neural networks (NNs) are widely explored and applied in recent years thanks to their abilities to handle highly complex users' behaviors. Different from MC-based methods, RNN-based technologies like HRNN [19] can model higher-order sequential dependencies while avoiding exponential growth of parameters existing in higher-order MCs [17]. However, RNN models suppose that items in a session follow a rigid order, which doesn't match the real-world session-based settings as a user might buy or look through these items randomly in a short time. Attention Mechanism assigns different weights to each part of the input and extracts more critical information. It makes more accurate judgments. MCRec [20] uses a deep neural network model with the co-attention mechanism to learn interaction-specific representations for users, items and meta-path context for top-N recommendation in HIN(heterogeneous information network). CoCoRec [21] leverages category information to capture the context-aware action dependence and uses a self-attention network to capture item-to-item transition patterns within each category-specific subsequence. The effectiveness of graph neural networks (GNNs) has been reported [22,23] in recommendation domains. MixGCF [24] utilizes the underlying GNN-based recommenders to synthesize negative samples. GHCF [25] uses GCN (Graph convolutional network) to explore the high-hop user-item interactions. MB-GMN [26] is an integrative neural architecture with a meta-knowledge learner and a meta graph neural network to capture the personalized multi-behavior characteristics. These models are usually trained in a pairwise manner, i.e., one item has priority over the

other. But in reality, items in the same session can not always have such partial order relations. Thus, these approaches might lead to false dependencies. Besides, RNN- and MC-based approaches are apt to forget long-term information and are biased to recently visited items according to their structures.

Recently, many SBRSs rely on NNs to model users' long and short-term interests. They pursue this task from two perspectives: (1) using an attention mechanism to learn explicit session-specific weights [7,9,27]; (2) exploiting different prototypes to model long and short-term sessions respectively [2,3,8]. Both of these two techniques have been proven very successful in SBRSs. Our work follows the first pipeline. However, these SBRSs treat each user independently and learn from users' personal behaviors to make recommendations, causing no explicit information sharing between similar users. Therefore, user and item representations are learned from a global view. In reality, there might be strong local associations inside users and items, which has been validated in many local NSRSs such as CMN [5] and r(s)GLSVD [14]. Unfortunately, NSRSs do not take temporal orders of user behaviors into account, which limits their performances. To this end, we propose a local SBRSs to combine the advantages of SBRSs and local fusion settings.

3 LSUG Model

3.1 Problem Formulation

In a recommender system (RS), we have a user u and an item v in a user set $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ and an item set $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$, respectively. Let $s = \{v_1, v_2, \cdots, v_{|s|}\} \subset \mathcal{V}$ be an item set clicked by a target user within a session, i.e. Δt. Throughout the history of user behaviors, we have a session sequence denoted by $\mathcal{S}_t^u = \{s_1^u, s_2^u, \cdots, s_t^u\}$ for each user, where t indicates the index of sessions following timestamps.

Formally, given a user u and his session sequence $\mathcal{S}_t^u$, we aim to build a model to predict the next items that have high probabilities belonging to current session s_t, by taking the consideration of user u's long and short-term sessions, as well as his long and short-term latent groups' influences.

3.2 Overview

Our model (the framework based on **L**ong and **S**hort-term latent **U**ser **G**roup modeling, LSUG) shown in Fig. 2 is a hierarchical end-end framework. It splits user behaviors to long and short-term ones. For each part, we aggregate item embeddings to form a user's preference representation. Then, based on the long and short-term representations, we calculate the probability distribution of groups that users might belong to, followed by an aggregation of group features to capture the impact from users' neighbors as well as the differences in preference between subsets of like-minded users. Finally, we construct a hybrid user representation using users' long and short-term session representations and group influences. Next, we give an introduction to each part of the model.

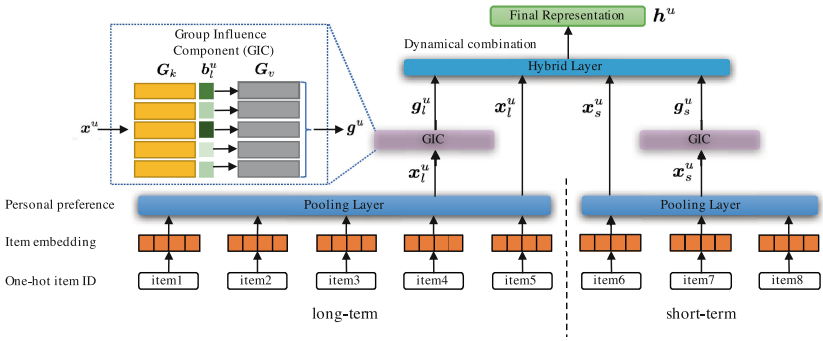


Fig. 2. The framework of LSUG.

3.3 General Embedding Construction

We use two matrices $\boldsymbol{U} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{V} \in \mathbb{R}^{M \times K}$ with fully-connected NNs to transform one-hot encoding of users and items to dense vectors, in which $N = |\mathcal{U}|$ (resp. $M = |\mathcal{V}|$) denotes the number of users (resp. items) and K is the latent dimensionality. Let $\boldsymbol{u} \in \mathbb{R}^K$ and $\boldsymbol{v} \in \mathbb{R}^K$ represent an embedding vector of user u and item v, respectively. They capture static features since they do not change with time.

Inspired by key-value memory networks (KV-MemNN) [28], to determine the influence of groups to each user, we assume L user groups with L latent anchor representations denoted by $\boldsymbol{G}_v \in \mathbb{R}^{L \times K}$. $\boldsymbol{G}_v$ describes the preferences of latent user groups. At the same time, latent groups have an additional representation matrix denoted by $\boldsymbol{G}_k \in \mathbb{R}^{L \times K}$ deciding the relations with users. $\boldsymbol{G}_k$ and $\boldsymbol{G}_v$ are similar to the *key* and *value* elements in KV-MemNN, respectively. In this way, we can assign a user to multiple groups and accumulate influences of user grouping for him.

3.4 Context-Aware Input Embedding

We split the sessions of a user into two parts and utilize his current session $s_t^u = \{v_1, v_2, \cdots, v_{|s_t^u|} | (u,t)\}$ to construct his context-aware input embeddings, which captures his **short-term** demands, and $s_{1:t-1}^u = \{v_1, v_2, \cdots, v_{|s_{1:t-1}^u|} | (u,t)\}$ as the **long-term** preference. Each session has a related embedding matrix which is computed according to the representations of the items in it.

Here, we feed these matrices to an aggregation function to learn semantic input embeddings $\boldsymbol{x}_s, \boldsymbol{x}_l \in \mathbb{R}^K$ as follows:

$$\begin{aligned} \boldsymbol{x}_s^u &= pooling(\boldsymbol{E}_t^u) \\ \boldsymbol{x}_l^u &= pooling(\boldsymbol{E}_{1:t-1}^u) \end{aligned} \tag{1}$$

We explore three different pooling methods to aggregate item features, i.e., mean, max, and attention pooling functions.

- *mean pooling:* It averages the values at each dimension of features. Each item has equal importance contributing to the final result.
- *max pooling:* It takes the max value for each dimension and captures item set features in an extreme way.
- *attention pooling:* It is a weighted average pooling, and we calculate the weights according to the relations between the general representation of the target user u and the item v as follows:

$$\begin{aligned} w_{u,v} &= \frac{exp(\boldsymbol{u}^\top \boldsymbol{v})}{\sum_{v_i \in s_t^u} exp(\boldsymbol{u}^\top \boldsymbol{v}_i)} \\ \boldsymbol{x}_t^u &= \sum_{v \in s_t^u} w_{u,v} \boldsymbol{v} \end{aligned} \tag{2}$$

The input embedding $\boldsymbol{x}_t^u$, where $t \in \{s, l\}$, reflects user u's status at different timestamps, such as his purchase requirements and related groups. Thus, it is reasonable to justify the relations between the groups, i.e., anchor points, and user u according to this input embedding $\boldsymbol{x}_t^u$.

3.5 Latent User Group Influence Modeling

We first calculate the similarity between the input embeddings, i.e., $\boldsymbol{x}_s^u, \boldsymbol{x}_l^u$, and *key* embeddings of latent groups $\boldsymbol{G}_k$, to assess a user's current probability distribution $\boldsymbol{b}_s^u, \boldsymbol{b}_l^u$ to decide which latent groups the user belongs to as

$$\begin{aligned} \boldsymbol{b}_s^u &= softmax(\boldsymbol{G}_k \boldsymbol{x}_s^u) \\ \boldsymbol{b}_l^u &= softmax(\boldsymbol{G}_k \boldsymbol{x}_l^u) \end{aligned} \tag{3}$$

where $\boldsymbol{b}_s^u, \boldsymbol{b}_l^u \in \mathbb{R}^L$ and we utilize $softmax(\cdot)$ function to convert the vector $\boldsymbol{G}_k \boldsymbol{x}^u$ to a pseudo probability distribution vector. Then, we aggregate group features, i.e., $\boldsymbol{G}_v$, according to the distributions to construct aggregated group feature vectors as:

$$\begin{aligned} \boldsymbol{g}_l^u &= \boldsymbol{G}_v \boldsymbol{b}_l^u \\ \boldsymbol{g}_s^u &= \boldsymbol{G}_v \boldsymbol{b}_s^u \end{aligned} \tag{4}$$

where $\boldsymbol{g}_l^u \in \mathbb{R}^K$ and $\boldsymbol{g}_s^u \in \mathbb{R}^K$ represent the long and short-term latent group influences to the user u, respectively.

3.6 Hybrid User Representation Modeling

This part yields a hybrid user representation from four aspects: two personal preference representations, i.e., users' long and short-term context-aware input session embeddings, and two group influence representations, i.e., the impacts from users' current groups and historical groups. To combine these four components in a dynamic way, we investigate two approaches to fuse them. For simplicity, we denote $\{\boldsymbol{x}_s^u, \boldsymbol{x}_l^u, \boldsymbol{g}_s^u, \boldsymbol{g}_l^u\}$ as $\boldsymbol{F}$.

– *MLP hybrid:* We use a Multi-Layer Perception (MLP) to map each feature vector to a scalar, followed by a softmax layer converting the scalar to a weight for each component.

$$\begin{aligned} w_{\boldsymbol{f}} &= \frac{exp(MLP(\boldsymbol{f}))}{\sum_{\boldsymbol{f'} \in \boldsymbol{F}} exp(MLP(\boldsymbol{f'}))} \\ \boldsymbol{h}^u &= \sum_{\boldsymbol{f'} \in \boldsymbol{F}} w_{\boldsymbol{f'}} \boldsymbol{f'} \end{aligned} \tag{5}$$

– *attention hybrid:* We calculate the weights according to the relations between the components and the embedding of the target user u.

$$\begin{aligned} w_{u,v} &= \frac{exp(\boldsymbol{u}^\top \boldsymbol{v})}{\sum_{v_i \in s_t^u} exp(\boldsymbol{u}^\top \boldsymbol{v}_i)} \\ \boldsymbol{x}_t^u &= \sum_{v \in s_t^u} w_{u,v} \boldsymbol{v} \end{aligned} \tag{6}$$

3.7 Model Learning

The total training procedure is shown in Algorithm 1. After the final user representation is learned, we compute the inner product of user representations and item embedding as their similarities or users' preferences to items:

$$\hat{R}_{u,v} = \boldsymbol{h}^{u\top} \boldsymbol{v} \tag{7}$$

We utilize a ranking and pairwise loss function proposed in [15] to train the model. For positive sampling, we randomly pick an item from user u's current session. And for negative sampling, we choose an item that the user u never bought or visited before. We denote the positive item and negative item as v^+ and v^-, respectively. Then, we calculate the final loss as follows:

$$\arg\min_{\Theta} \sum_{(u, S_t^u, v^+, v^-) \in \mathcal{D}} -\ln \sigma(\hat{R}_{u,v^+} - \hat{R}_{u,v^-}) \tag{8}$$

where $\mathcal{D}$ is the train set containing all samples, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function.

4 Experiments

4.1 Experimental Setup

Datasets. We use Tmall dataset [29], which contains purchase behaviors of users on Tmall online shop, and Gowalla dataset [30], which collects check-in behaviors of users. Following the settings in [9], we keep the last seven months of

Algorithm 1: Traning process

input : embedding dimension K, number of group L, sessions data S, initial learning rate η
output: trained model with parameters Θ
do initialization;
shuffle the sessions data S
while *not convergence* **do**
  for *batch in* S **do**
    randomly select t' for each session sequence and split sessions to long and short-term;
    do positive sampling in last session;
    do negative sampling in unvisited items;
    compute loss according to Eq. (8);
    do backpropagation and update parameters Θ;
  end
end

data and items that have been observed by no less than 20 users. We aggregate items purchased in one day by the same user into a session and remove the sessions that only contain a single item. We randomly pick 20% of total users for test and randomly select an item in their last session as the target item to be predicted. Then, the statistics of datasets are shown in Table 1.

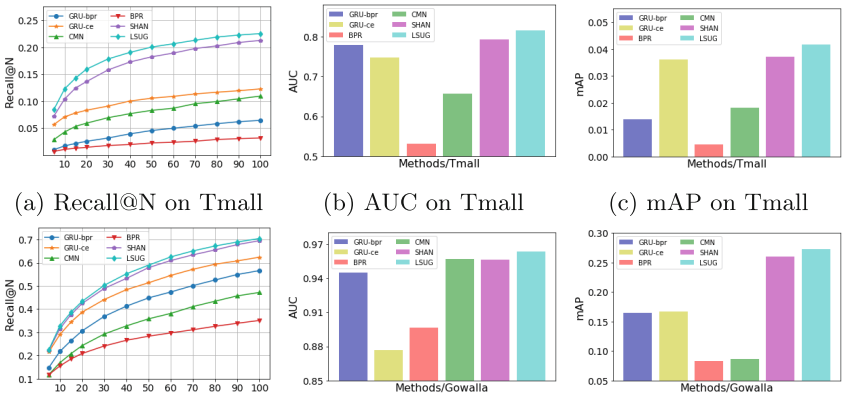
Table 1. Statistics of datasets

Dataset	Tmall	Gowalla
#user	20202	15076
#item	24774	12419
avg. session length	2.72	2.95
#train session	70895	128374
#test session	4040	3015
user-item matrix density	0.039%	0.15%

Baselines. We compare our model with the following baselines, including SBRSs, NSRSs, local NSRSs. 1) **BPR** [15] is a classic NSRS that learns how to rank from users' feedback data by pairwise optimization. 2) **GRU4Rec-bpr** and 3) **GRU4Rec-ce** [12] are outstanding algorithms that uses GRU (gated recurrent unit) to model the sequential data. The former uses BPR as the ranking loss function while the latter takes cross-entropy as the loss function. 4) **CMN** [5] is a kind of local NSRSs that takes users' neighbors as the values in the memory bank. 5) **SHAN** [9] is a state-of-the-art SBRS, which also utilizes a hierarchical neural network.

Metrics. We use Recall, AUC, and mAP to evaluate models. Recall measures how much the prediction covers the ground truth. AUC evaluates how highly positive examples have been ranked over negative examples. mAP evaluates the location of the real visited items in the predicted list.

Parameters Settings. We set K to 150 and L to 512 for both datasets. The initial learning rate is set to 0.03 with a 0.8 decay rate in every eight steps. We train the model until the convergence is reached. In the final models, we choose {*pooling layer: mean pooling*} and {*hybrid layer: MLP hybrid*} for Tmall dataset, while choose {*pooling layer: attention pooling*} and {*hybrid layer: attention hybrid*} for Gowalla dataset since they perform best in our experiments.



(a) Recall@N on Tmall (b) AUC on Tmall (c) mAP on Tmall

(d) Recall@N on Gowalla (e) AUC on Gowalla (f) mAP on Gowalla

Fig. 3. Performance comparison

4.2 Comparison of Performance

Fig. 3 shows the performance of LSUG and other baselines on both Tmall and Gowalla datasets under all metrics. From the figure, we can observe that:

1) Our LSUG outperforms all the baselines, including a latent factor CF model, i.e., BPR, a local NSRS, i.e., CMN, two sequential models, i.e., GRU-bpr and GRU-ce, and a hierarchical SBRS, i.e., SHAN, with a large margin, especially on Tmall dataset. For example, LSUG improves 16.9% compared with SHAN (15.9% v.s. 13.6%) at Recall@20 and 4.16% at Recall@100 (22.5% v.s. 21.6%), although SHAN outperforms other baselines. Since both LSUG and SHAN split the sessions into long and short-term ones as well as both of them are a hierarchical model, the gain of performance might come from **group influence components (GICs)**, which indicates the GICs are beneficial to model user preferences and help to make better recommendations.

2) Both GRU4Rec-bpr and GRU4Rec-ce perform well, the reason is that they might successfully capture sequential patterns, i.e., the dependency relation between items. Moreover, GRU-ce is better than GRU-bpr under Recall@N and AUC metrics. The reason might be that the softmax layer computes the probability of positive items over all negative ones, while the bpr loss only uses the sampled item pairs.
3) Although CMN and BPR are both NSRSs, CMN outperforms BPR. It may be because CMN collects user neighbors' preferences for the current user. It further proves local information helps model user preferences.

Table 2. Influence of pooling functions and hybrid methods

Dataset	Tmall		Gowalla	
pooling-hybrid	recall@20	mAP	recall@20	AUC
SHAN	0.136	0.037	0.424	0.956
max-attn	0.146	0.045	0.412	0.957
max-MLP	0.136	0.039	0.399	0.944
mean-attn	0.153	**0.047**	0.429	0.961
mean-MLP	**0.159**	0.042	0.424	0.957
attn-attn	0.141	0.041	**0.433**	**0.962**
attn-MLP	0.155	0.042	0.400	0.954

4.3 Influence of Components

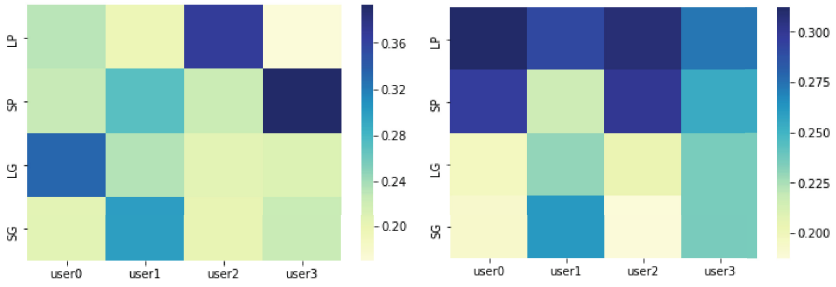
Influence of Latent User Groups. To further investigate the effectiveness of user groups, we remove the group features from the model and only combine $\boldsymbol{x}_s^u$ and $\boldsymbol{x}_l^u$. It leads to the results shown in Fig. 5, in which LSUG-d denotes LSUG deleting group features. We could see that, without group features, the performance of the model becomes worse, which proves that the group features are important.

Influence of Pooling and Hybrid Methods. To show the influence of aggregation methods, we exhibit the performance of all combinations of {*pooling layer: mean pooling, max pooling, attention pooling*} × {*hybrid layer: attention hybrid, MLP hybrid*}. As shown in Table 2, for combining item features, i.e., the pooling layer in our model, mean pooling is better than max pooling under all experimental settings. e.g., mean-MLP v.s. max-MLP. The reason might be that mean pooling takes all item features into consideration and passes the information to the downstream network, while max pooling only picks the most extreme features. Attention pooling sometimes obtains worse results than mean pooling. A possible explanation is that sometimes the target item is not similar to a

user's general preference representation, i.e., $\boldsymbol{u}$, and thus the model pays false attention.

For hybrid methods, i.e., the hybrid layer in our model, attention and MLP get comparable results, and we couldn't conclude that which one is better. However, most combinations achieve better results than SHAN steadily on both datasets, which show the power of user group modeling. On Tmall dataset, attn-MLP outperforms attn-attn with a large margin, while on Gowalla dataset, the observation is the opposite. We note that Gowalla dataset records users' check-in data, and a user could visit one place repeatedly, while a user purchases already bought items with much lower frequency on Tmall dataset. Under such a circumstance, user embeddings on Gowalla dataset might be more similar to the frequently visited items. These items occur a lot in the test set as well. As a result, on Gowalla dataset, attention mechanism considers general user preferences, leading to better performance. On the contrary, MLP only considers current features and thus gets worse results on Gowalla dataset but yields a better performance on Tmall dataset.

We randomly sample several users from both datasets and visualize their weights of long and short-term personal preferences (i.e., LP & SP), and long and short-term group influences (i.e., LG & SG), as shown in Fig. 4. We can observe that the weights are customized for different users.



(a) Tmall (b) Gowalla

Fig. 4. Weights visualization

4.4 Influence of Hyper-Parameters

We study the influence of latent group size L in our model. The value of L changes from 100 to 1000, and we only record Recall@20 values due to the limited space. As shown in Fig. 6, as group size grows, the value of metric increases gradually at first on both datasets. This indicates that small size is not enough to cover all potential latent user groups. However, the too larger size also decreases the performance because it might cause overfitting problems. Above all, a proper group size, e.g., $L = 500$ on Tmall dataset and $L = 600$ on Gowalla dataset, should be tuned to gain remarkable results.

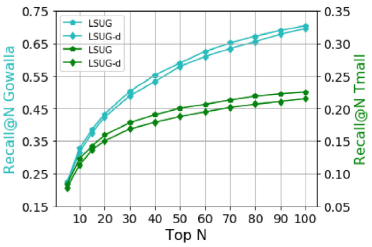


Fig. 5. The performance of LSUG and LSUG-d

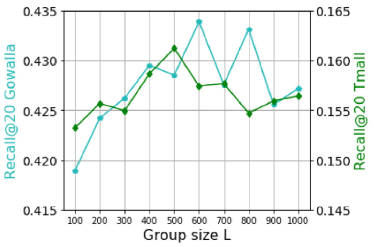


Fig. 6. The influence of group size L

5 Conclusion

In this paper, we proposed a next-item recommendation model based on learning long and short-term user groups. Specifically, we split user behaviors into long and short-term sessions. For all sessions, we abstract their representations according to their items. After that, the designed GICs detect users' latent long and short-term groups, and incorporate the influences from different latent groups to form the final user representations. The conducted experiments on two real-world datasets demonstrate that our model outperforms several state-of-the-art models in terms of multiple metrics.

Acknowledgments. This work is supported by Shanghai Youth Science and Technology Talents Sailing Program (No. 22YF1413700). Thanks to Runtong Li and Xingjing Lu for their valuable advice and help.

References

1. Yu, L., Zhang, C., Liang, S., Zhang, X.: In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5709–5716 (2019)
2. Guo, L., Yin, H., Wang, Q., Chen, T., Zhou, A., Hung, N.Q.V.: In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1569–1577. ACM (2019)
3. Zhao, W., Wang, B., Ye, J., Gao, Y., Yang, M., Chen, X.: In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 3676–3682 (2018)
4. Wang, S., Hu, L., Wang, Y., Sheng, Q.Z., Orgun, M., Cao, L.: In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1–7. AAAI Press (2019)
5. Ebesu, T., Shen, B., Fang, Y.: In: Proceedings of the International SIGIR Conference on Research & Development in Information Retrieval, pp. 515–524. ACM (2018)
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: In: Proceedings of the International Conference on World Wide Web (WWW) (International World Wide Web Conferences Steering Committee, pp. 173–182 (2017)

7. Zhu, N., Cao, J., Liu, Y., Yang, Y., Ying, H., Xiong, H.: In: Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM), pp. 807–815. ACM (2020)
8. Zhang, S., Tay, Y., Yao, L., Sun, A., An, J.: In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 9 (2019)
9. Ying, H., et al.: In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2018)
10. Zhu, N., Cao, J., Lu, X., Xiong, H.: ACM Trans. Inf. Syst. (TOIS) **40**(2), 1 (2021)
11. Koren, Y., Bell, R., Volinsky, C.: Computer (8), 30 (2009)
12. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: In: 4th International Conference on Learning Representations, ICLR (2016)
13. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 346–353 (2019)
14. Christakopoulou, E., Karypis, G.: In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1235–1243. ACM (2018)
15. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: CoRR abs/1205.2618 (2012)
16. Li, W., et al.: IEEE Access **7**, 45451 (2019)
17. He, R., McAuley, J.: In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 191–200. IEEE (2016)
18. Zhang, R., Mao, Y.: IEEE Access **7**, 13189 (2019)
19. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: In: Proceedings of the Eleventh ACM Conference on Recommender Systems (Association for Computing Machinery, New York, NY, USA, RecSys 2017, pp. 130–137 (2017)
20. Hu, B., Shi, C., Zhao, W.: In: The 24th ACM SIGKDD International Conference (2018)
21. Cai, R., Wu, J., San, A., Wang, C., Wang, H.: In: SIGIR 2021: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021)
22. Ge, S., Wu, C., Wu, F., Qi, T., Huang, Y.: In: Proceedings of The Web Conference 2020, New York, NY, USA, pp. 2863–2869. Association for Computing Machinery (2020)
23. Zheng, J., Ma, Q., Gu, H., Zheng, Z.: In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, New York, NY, USA, KDD 2021, pp. 2338–2348. Association for Computing Machinery (2021)
24. Huang, T., et al.: In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD 2021, New York, NY, USA, pp. 665–674. Association for Computing Machinery (2021)
25. Chen, C., et al.: In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3958–3966 (2021)
26. Xia, L., Xu, Y., Huang, C., Dai, P., Bo, L.: In: SIGIR 2021: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021)
27. Yuan, J., Song, Z., Sun, M., Wang, X., Zhao, W.X.: In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4635–4643 (2021)
28. Miller, A.H., Fisch, A., Dodge, J., Karimi, A., Bordes, A., Weston, J.: In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1400–1409 (2016)
29. Hu, L., Cao, L., Wang, S., Xu, G., Cao, J., Gu, Z.: In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1858–1864 (2017)

30. Cho, E., Myers, S.A., Leskovec, J.: In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1082–1090. ACM (2011)