Sequential Recommendation in Online Games with Multiple Sequences, Tasks and User Levels

Si Chen*
sichen@stu.xmu.edu.cn
School of Informatics, Xiamen University
Xiamen, China

Yuqiu Qian
yuqiuqian@tencent.com
Tencent
Shenzhen, China

Hui Li†
hui@xmu.edu.cn
School of Informatics, Xiamen University
Xiamen, China

Chen Lin†
chenlin@xmu.edu.cn
School of Informatics, Xiamen University
Xiamen, China

ABSTRACT

Online gaming is growing faster than ever before, with increasing challenges of providing better user experience. Recommender systems (RS) for online games face unique challenges since they must fulfill players' distinct desires, at different user levels, based on their action sequences of various action types. Although many sequential RS already exist, they are mainly single-sequence, single-task, and single-user-level. In this paper, we introduce a new sequential recommendation model for multiple sequences, multiple tasks, and multiple user levels (abbreviated as M^{3}Rec) in Tencent Games platform, which can fully utilize complex data in online games. We leverage Graph Neural Network and multi-task learning to design M^{3}Rec in order to model the complex information in the heterogeneous sequential recommendation scenario of Tencent Games. We verify the effectiveness of M^{3}Rec on three online games of Tencent Games platform, in both offline and online evaluations. The results show that M^{3}Rec successfully addresses the challenges of recommendation in online games, and it generates superior recommendations compared with state-of-the-art sequential recommendation approaches.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

online games, sequential recommender systems, multi-task learning, graph neural network

*Work done when the first author was an intern in Tencent.
†Hui Li and Chen Lin are the corresponding authors. Email: hui@xmu.edu.cn, chenlin@xmu.edu.cn.

SSTD '21, August 23–25, 2021, virtual, USA

ACM ISBN 978-1-4503-8425-4/21/08…$15.00
https://doi.org/10.1145/3469830.3470906

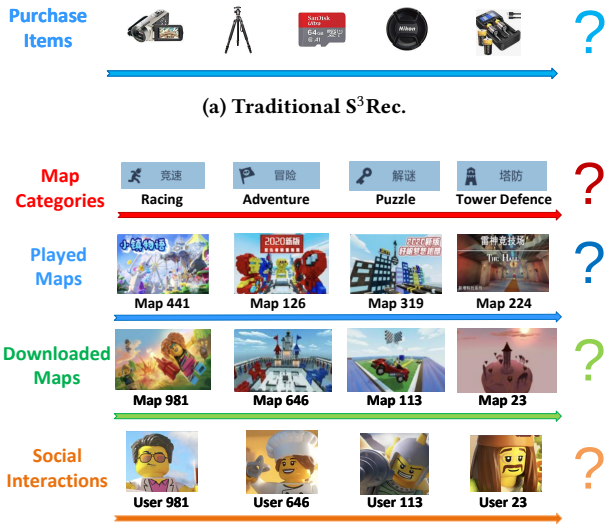
ACM Reference Format:
Si Chen, Yuqiu Qian, Hui Li, and Chen Lin. 2021. Sequential Recommendation in Online Games with Multiple Sequences, Tasks and User Levels. In *17th International Symposium on Spatial and Temporal Databases (SSTD '21), August 23–25, 2021, virtual, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3469830.3470906

1 INTRODUCTION

As web services are ever-expanding, sequential data (e.g., users' click logs or user traveling history) become prevalent in Recommender Systems (RS) and therefore *Sequential Recommender Systems* (SRS) have attracted more and more attention [11, 35, 43]. Given users' historical behavior sequences, SRS aim at predicting his/her next action, e.g., the next point of interest (i.e., POI) to visit, or the next product he/she will buy. Unlike traditional RS which learn from the *two-way* user-item interactions, SRS model the *three-way* interaction among a user, an item he/she has selected and an item he/she will select next. One specific application of SRS is recommendation in online games which is growing faster than ever before, with increasing challenges of providing better user experience. In this paper, we study the sequential recommendation problem in *Tencent Games*[1] where the heterogeneous information makes it difficult for traditional SRS to fully model the sequential behaviors of users.

Although many SRS already exist [11, 35, 43], they are mainly single-sequence, single-task, and single-user-level (i.e., modeling individual user instead of user groups), i.e., *single-task learning at single-user level for single-sequence RS* (S^{3}Rec for short). Figure 1a demonstrates an example of such SRS. The blue timeline indicates the purchase sequence, i.e., products that one user bought. The task is to recommend the next item to purchase for the user. As the previous items are related to video cameras, the next item is likely to be also related to video camera. Some recent methods consider more content features (e.g., temporal feature, text and item category) [17, 57] to enhance SRS's ability. Though heterogeneous information has been considered, these methods still model sequential behaviors based on a single sequence. Additional information is utilized as auxiliary data and incorporated into the single sequence to optimize for a single task, i.e., predicting the next item the user will interact with.

[1]https://game.qq.com



(a) Traditional S^{3}Rec.

(b) M^{3}Rec for LEGO®Cube.

Figure 1: A comparison between traditional S^{3}Rec and M^{3}Rec for Tencent online game LEGO®Cube.

S^{3}Rec is suitable for recommendation scenarios such as book or movie recommendation. However, the recommendation in online games faces unique challenges that S^{3}Rec cannot handle well:

- Firstly, the action types are diverse in online gaming platforms. In other recommendation scenarios, such as book or movie recommendation, users can only adopt limited actions, e.g., purchase or click. However, in online games, players are able to adopt many types of actions. For example, Figure 1b depicts part of the recommendation scenario of the game LEGO®Cube[2] in Tencent Games platform, where heterogeneous information (e.g, map category, played maps, downloaded maps, and social interactions) is contained in *multiple user action sequences* and they have different meanings for the same user. Conventional SRS are unable to distinguish such differences.
- Secondly, there are multiple tasks to be fulfilled in online gaming platforms. In other recommendation scenarios, recommendation providers are only interested in the main action type, e.g., purchase. However, predicting the next item to purchase is not the only task in online games. For example, in LEGO®Cube, the *multiple tasks* of predicting next map to download, next map to play are all important for the game developer and publisher to efficiently allocate resource, improve user experience, and increase revenue.
- Finally, there is different representation levels for users and user groups in online gaming platforms. Most online games incorporate strong social factors that allow users to team up in battles and regions. Thus, in order to deliver more accurate recommendations, SRS for online games need to model not only single-level user representations (i.e., representations for each single user) but also *multiple user-level* representations (i.e., representations for user groups). However, few existing works consider modeling user representations in sequential recommendation, as pointed out by Fang et al. [11].

The new challenges in SRS for Tencent Games and the limitations of previous SRS motivate us to propose a *sequential recommender system with multiple sequences, tasks and user levels* (abbreviated as M^{3}Rec). We leverage Graph Neural Network (GNN) [67] and multi-task learning [70] to design a multi-sequence, multi-task and multi-level neural recommendation architecture for M^{3}Rec to fully utilize the complex information in the heterogeneous sequential recommendation scenario of Tencent Games. More descriptions for the recommenders in Tencent Games are provided in Section 2.

In summary, the major contributions of this paper are:

- We identify and formally define a new multi-sequence, multi-task and multi-level sequential recommendation problem, which is a practical problem existing in online games of Tencent Games platform. To our best knowledge, we are first to study this problem in terms of sequential recommendation in online games.
- We propose a new framework M^{3}Rec for sequential learning in the heterogeneous recommendation scenario. M^{3}Rec has the capacity to model the complex meanings of multiple sequences beneath the surface.
- We adopt multi-task learning in M^{3}Rec so that it is able to optimize several tasks in parallel.
- We evaluate M^{3}Rec on real data of three online games in Tencent Games platform and verify the effectiveness of M^{3}Rec for the traditional task of recommending next item in SRS. Meanwhile, M^{3}Rec can offer suggestions for other prediction tasks in online games with high quality.

The rest of the paper is organized as follows: Section 2 gives an overview of the recommendation scenario in Tencent Games and formally defines the new recommendation problem. We illustrate our proposed method, M^{3}Rec, in Section 3. In Section 4, we compare M^{3}Rec with other state-of-the-art sequential recommendation algorithms and verify its effectiveness. Section 6 concludes our work.

2 SEQUENTIAL RECOMMENDER SYSTEM FOR LEGO®CUBE

In this section, we will give an overview of LEGO®Cube as an example to illustrate the recommendation scenario of online games in Tencent Games platform. Then, we will define the sequential recommendation problem for M^{3}Rec in online games.

2.1 Overview of SRS for LEGO®Cube

LEGO®Cube is a popular sandbox game developed and operated by Tencent Games. It provides a virtual world comprised of different areas for Chinese mobile players. Players are encouraged to freely download area maps, explore these areas, construct with LEGO bricks, and get involved in various tasks with friends. As such, for each player, four behavior sequences are available: the *map sequence* containing game maps that each user has downloaded, the *type category* containing map categories of each user's downloaded

[2]https://lgwx.qq.com

maps, the *play sequence* containing game maps played by each player, and the *friend sequence* containing other users that each user has interacted with.

Figure 1b illustrates how the SRS in LEGO®Cube organizes the data of multiple sequences for one player:

- The green timeline contains the downloaded map sequence of this player. It corresponds to the main task, namely recommending the next map that the target player will download.
- The red timeline contains the map categories corresponding to maps in the green timeline. The task for this sequence is to predict the map category of the next map to be recommended. This task is helpful for game operation engineers when the information of some players is insufficient to make fine-grained recommendations and they can use the category recommendation instead of map recommendation.
- The blue timeline contains the played map sequence of this player. The auxiliary task for this sequence is to predict which map will be played by the player next. As maps must be firstly downloaded and then they can be played, this task is useful for distinguishing maps which players will play even without being recommended in the download sequence (i.e., main task) and maps that players will play after they get exposure via the recommendation. Game operation engineers may tune the recommendation priority for the latter in the main task so that they can be played.
- The orange timeline contains the user-user interaction sequence of this player. The auxiliary task for this sequence is to predict the next player that this player will interact with. The predication results of this sequence can help game operation engineers identify similar or relevant players, which is used as part of the input to the game player clustering in Section 3.3.

The meanings of the elements in each sequence are different under the surface. For example, one map appearing in green timeline indicates that the player has downloaded the map, but it is not the indication of his/her preferences. Since SRS in the game may push maps to players based on some strategies (e.g., advertising), it is possible that players will download some maps that he/she does not feel interesting actually. As a comparison, when a player played a map in blue timeline, the system can understand that it is a strong and explicit indication of his/her preferences. In summary, there are five recommendation problems which naturally arise from the sequences shown in Figure 1b:

(1) **Next map prediction** is the main task that predicts which map will be downloaded by the user next.
(2) **Next category prediction** is an auxiliary task that predicts which category of map will be downloaded by the user next.
(3) **Next play prediction** is an auxiliary task that predicts which map will be played by the user next.
(4) **Next friend prediction** is an auxiliary task that predicts who will be added to the user's friend list next.
(5) **Next group map prediction** is an auxiliary task that predicts the next download map for a user group.

The last task is related to a concept of *multiple user levels* in online games. Game operation engineers typically group users by group characters and sometimes promote game activities to some specific user group. In this case, the recommendation is perform on the second user level, i.e., user group, instead of the first user level, i.e., individual player.

Different sequences in LEGO®Cube involve different meanings of players' behaviors and simply incorporating them into one sequence without distinction will result in information loss. On the other hand, prediction tasks in LEGO®Cube vary in their goals and optimizing them simultaneously in one sequence and single user level is not straightforward. Therefore, previous sequential recommendation algorithms, which study single sequence, task and user level, cannot model the complex scenario in LEGO®Cube.

2.2 Problem Definition

We now first give the formal definition of the recommendation task of S^3Rec:

Definition 2.1 (Recommendation Task of S^3Rec). Let $I = \{i_1, \dots, i_{|I|}\}$ be a set of items. The action sequence s for a user is the sequence of items that the user has interacted with: $\{a_1, \dots, a_{|s|}\}$, where $a_j = \langle i_j, t_j \rangle$ indicates that the user selected item i_j at time t_j with $t_{|s|} \geq t_{|s|-1} \geq \dots \geq t_1$, and $|s|$ is the number of actions in s. Given an action sequence s, S^3Rec predicts the item(s) that the user will next add to s. Items can be replaced by users or item categories if the sequence is related to user-user interactions or item information like categories.

We can extend Definition 2.1 and define the recommendation task of M^3Rec:

Definition 2.2 (Recommendation Task of M^3Rec). Let $I = \{i_1, \dots, i_{|I|}\}$ be a set of items. The action sequence set $\mathcal{S} = \{s_1, \cdots, s_m\}$ for a user (or a user group) has m sequences (one main sequence and $m-1$ auxiliary sequences) with possible different lengths. Each sequence s_q consists of items that the user (or user group) has interacted with: $\{a_1, \dots, a_{|s_q|}\}$, where $a_j = \langle i_j, o, t_j \rangle$ indicates that the user (or user group) performed action o on item i_j at time t_j, and $t_{|s_q|} \geq t_{|s_j|-1} \geq \dots \geq t_1$. Given $\mathcal{S}$, the main task of M^3Rec is to predict the item in the next action that the user (or user group) will next add to the main sequence. Other auxiliary tasks aim at predicting the item/item category/user in the next action that the user (or user group) will next add to the auxiliary sequences.

Similar to S^3Rec, M^3Rec does not predict the timestamp for the next action. Besides, the action type o is the same for all the actions in one sequence. Therefore, M^3Rec does not need to predict the action type for new action.

3 M^3REC FOR TENCENT GAMES PLATFORM

In this section, we will first review the Gated Graph Sequence Neural Network [29] based S^3Rec [66] since M^3Rec is built on the top of it. Then we will describe how M^3Rec is able to model the sequential recommendation problem with multiple sequences, tasks and user level which is the scenario that online games in Tencent Games Platform have. Figure 2 provides an overview of M^3Rec.

3.1 GGS-NN Based S^3Rec

Li et al. [29] introduce the gate mechanism, which has been shown to be effective in Long Short-term Memory, into Graph Neural

Network (GNN) [48] and propose the Gated Graph Sequence Neural Network (GGS-NN) for modeling graphs as sequences. Later, Wu et al. [66] apply GGS-NN to S^{3}Rec. In M^{3}Rec, we use their design for the single-sequence, single-task and single-user-level recommendation unit in M^{3}Rec, i.e., each of the tasks 1 to N in Figure 2.

Specifically, each user action sequence s can be viewed as a sequence graph $\mathcal{G}_s$ and the update rule for the representation h_i of each node (i.e., item) i in s from epoch $t-1$ to epoch t using a GGS-NN layer can be defined as follows:

$$\begin{aligned}
\mathbf{c}_{s,i}^{(t)} &= \mathbf{A}_{s,i:}^T \left[\mathbf{h}_1^{(t-1)}, \cdots, \mathbf{h}_n^{(t-1)}\right]^T + \mathbf{b} \\
\mathbf{z}_{s,i}^{(t)} &= \sigma(\mathbf{W}_z \mathbf{c}_{s,i}^{(t)} + \mathbf{V}_z \mathbf{h}_i^{(t-1)}) \\
\mathbf{r}_{s,i}^{(t)} &= \sigma(\mathbf{W}_r \mathbf{c}_{s,i}^{(t)} + \mathbf{V}_r \mathbf{h}_i^{(t-1)}) \\
\tilde{\mathbf{h}}_i^{(t)} &= tanh\big(\mathbf{W}_h \mathbf{c}_{s,i}^{(t)} + \mathbf{V}_h (\mathbf{r}_{s,i}^{(t)} \odot \mathbf{h}_i^{(t-1)})\big) \\
\mathbf{h}_i^{(t)} &= (1 - \mathbf{z}_{s,i}^{(t)}) \odot \mathbf{h}_i^{(t-1)} + \mathbf{z}_{s,i}^{(t)} \odot \tilde{\mathbf{h}}_i^{(t)}
\end{aligned} \tag{1}$$

where $\mathbf{c}$, $\mathbf{z}$, $\mathbf{r}$ and $\tilde{\mathbf{h}}$ are gathering gate, reset gate, update gate, and candidate activation, respectively. $\sigma(\cdot)$ is the sigmoid function, $\mathbf{h}_*$ is the vector representation of node in sequence s, "$\odot$" is the element-wise multiplication, and $\mathbf{W}_*$ and $\mathbf{V}_*$ are learnable matrices. $\mathbf{A}_s \in \mathbb{R}^{n_s \times 2n_s}$ is the connection matrix for s (i.e., the concatenation of outgoing adjacent matrix and ingoing adjacent matrix), and n_s indicates the number of unique nodes in s. $\mathbf{A}_{s,i:} \in \mathbb{R}^{n_s \times 2}$ are the two columns of blocks in $\mathbf{A}_s$ with respect to node i. For instance, Figure 3 illustrates an example showing the connection matrix for one user's user-item interaction sequence $\{v_1, v_2, v_3, v_2, v_4\}$.

We may stack several GGS-NN layers to enhance the non-linearities of the model and we call it *GGS-NN unit* in the following. For one user's user-item interaction sequence $\{v_1, \cdots, v_l\}$ with l items in the sequence, the prediction for the next item is generated based on the local sequence embedding (i.e., the learned representation $\mathbf{h}_l$ for the last node v_l in the sequence) and the global sequence embedding $\mathbf{s}_g$ which is captured by a soft-attention layer:

$$\alpha_i = \mathbf{g}^T \sigma(\mathbf{W}_1 \mathbf{h}_l + \mathbf{W}_2 \mathbf{h}_i + \mathbf{c}), \quad \mathbf{s}_g = \sum_{i=1}^{l} \alpha_i \mathbf{h}_i \tag{2}$$

where $\mathbf{g}$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weights. $\mathbf{b}$ is the bias vector.

Finally, the predicted probability of an item j being the next item is:

$$\hat{r}_j = \big(\mathbf{W}_3(\mathbf{s}_g \oplus \mathbf{h}_l)\big)^T \mathbf{v}_j, \tag{3}$$

where $\mathbf{v}_j$ is the embedding for the item j, $\mathbf{W}_3$ is a learnable weight matrix, and "$\oplus$" indicates the concatenation operation.

3.2 Modeling Multiple Sequences and Multi Tasks in M^{3}Rec

Considering the multi-sequence and multi-task scenario that online games face, we choose to adopt multiple GGS-NN units for online games in Tencent Game platform. Each of them is used for a specific sequence and a specific task. Items in the original GGS-NN unit are replaced with item categories or users according to the task. For next category prediction task, a separate map category embedding layer is used instead of the item embedding layer.

As shown in Figure 3, each task is modeled using an independent GGS-NN unit with the soft-attention mechanism, and all tasks share the user embedding layer and item embedding layer. For instance, the same user i in the next map prediction task and the next friend prediction task of LEGO®Cube will share the same user embedding $\mathbf{u}_i$ encoded by the shared user embedding layer. All the GGS-NN unit are trained together in the manner of multi-task learning. The details of the optimization will be illustrated in Section 3.4.

3.3 Modeling Multiple User Levels in M^{3}Rec

Game operation engineers lead game players to particular game activities for a better user experience, and a game activity typically has a target user group. Promoting game activities may increase the chance for users to see the maps they favor. Game players in LEGO®Cube have group characters, and operations engineers regularly update the user statistics. We run k-means clustering algorithm over these data to construct the player grouping information.

After identifying the player groups in the game, we are actually dealing with a classical problem in the recommender system community, namely *group recommendation* [2, 47], in the sequential recommendation task in Tencent Games platform. In traditional group recommendation task, the system aims at recommending items to a group of users that maximize the satisfaction of the group members, according to some semantics of group satisfaction. Users in the same group share some similarities (e.g., tastes or locations), and all members in the same group receive the same recommendation from the system. Group recommendation has been widely studied and applied in social recommenders (e.g., Shelfari [27]), event recommenders (e.g., Plancast [34]), restaurant recommenders (e.g., ZAGAT [39]), just to name a few. However, only a few studies [40, 54] has considered the group recommendation problem in sequential recommender.

We first define the *Hierarchical Multi-level Recommendation* for online games. Formally, we have a two-level recommendation architecture as shown in Task $N+1$ of Figure 2. On the bottom level, M^{3}Rec conducts the general single-level sequential recommendation task, i.e., recommend an item to each individual game player. On the second level, M^{3}Rec turns to *sequential group recommendation* task, i.e., recommend an item to groups of game players.

We design an attention based aggregation layer to aggregate the information from each group member at the bottom level and then construct the representation of the group on the second level. Recall that $\mathbf{h}_i$ indicates the representation of item i. Suppose that $\mathbf{q}_j$ is the representation of player j on the first level, $\mathbf{p}_g$ is the representation of player group g at the bottom level, and $\mathcal{N}(g)$ indicates all the players that player group g contains. We aggregate the representations of individual players and generate the embedding for group g on the second level:

$$\begin{aligned}
\mathbf{p}_g &= \mathrm{RELU}\big(\mathbf{W}_p \sum_{i \in \mathbf{M}_g} \beta_i \mathbf{h}_i\big) \\
\beta_i &= \frac{exp(\hat{\beta}_i)}{\sum_{i' \in \mathcal{N}(g)} exp(\hat{\beta}_{i'})} \\
\hat{\beta}_i &= \mathbf{W}_b \mathbf{e}_i + b_e \\
\mathbf{e}_i &= \mathrm{RELU}\big(\mathbf{W}_e \mathbf{h}_i + \mathbf{b}_e\big)
\end{aligned} \tag{4}$$

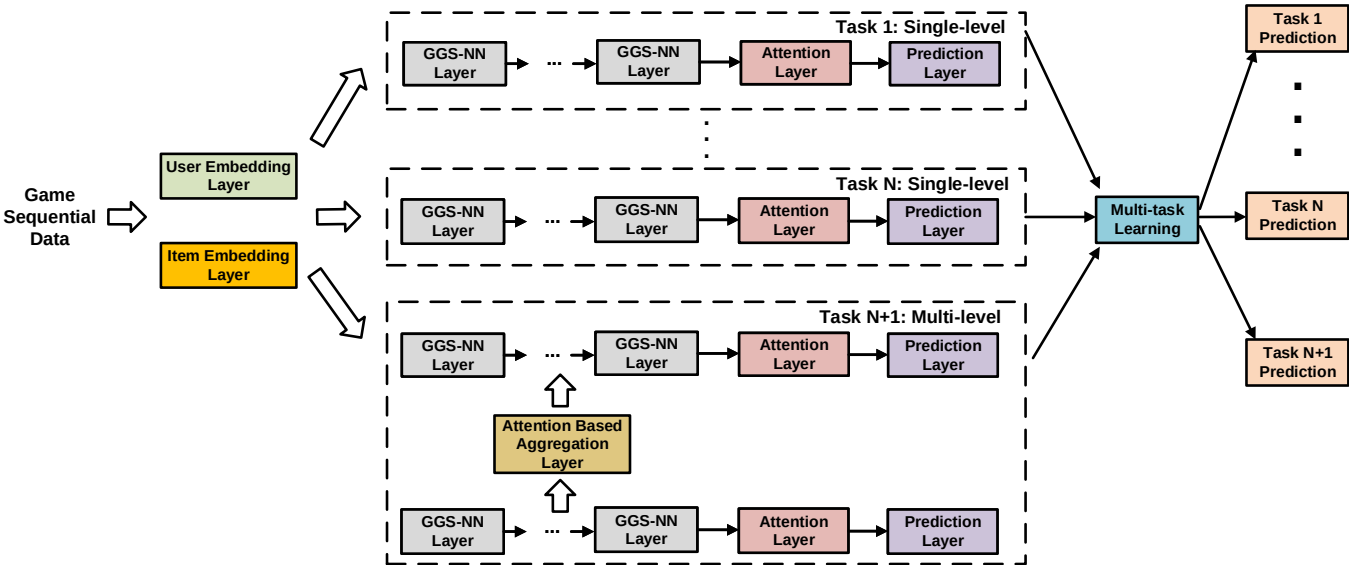


Figure 2: Overview of M^{3}Rec for online games.

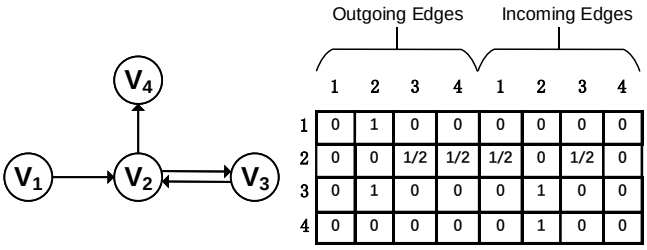


Figure 3: An example of a sequence graph and the connection matrix A.

where $RELU(\cdot)$ is the Rectified Linear Unit, $\mathbf{W}_*$ is a weight matrix, $\mathbf{b}_e$ is a bias vector, and b_e is a bias term. In Equation 4, we assign different attention weight β to indicate the differing importance of each group member to the group. The attention weight β is parameterized with a single-layer feedforward neural network and then normalized by the softmax function. The motivation is that differences still exist among game players belonging to the same group, though players in the same group share very similar characteristics. Considering such nuances helps model the profile of the whole group better.

After obtaining the group representations $\mathbf{p}$, the group recommendation process at the second level is similar to the process of GGS-NN based S^{3}Rec illustrated in Section 3.1.

3.4 Optimization of Multi-task Learning

For each individual task, the predicted probability distribution for each possible item/item category/user being the next in an action sequence is similar to Equation 3:

$$\hat{\mathbf{r}} = \left(\mathbf{W}_3(\mathbf{s}_g \oplus \mathbf{h}_l)\right)^T \mathbf{Q}, \tag{5}$$

where $\mathbf{Q}$ is all the item embeddings, all the item category embeddings, or all the user embeddings, depending on the detailed task.

Then, the cross-entropy loss is used in the optimization for each task i:

$$\mathcal{L}_i = -\sum_{s \in \mathcal{S}_i} \mathbf{r}_s \log(\hat{\mathbf{r}}_s), \tag{6}$$

where $\mathcal{S}_i$ is the interaction sequence set for task i, $\mathbf{r}_s$ indicates the one-hot encoding vector of the ground-truth next item/item category/user in the sequence s, and $\hat{\mathbf{r}}_s$ is calculated using Equation 5.

We use a common method to perform multi-task learning in M^{3}Rec, i.e., assign a task weight to each task loss and optimize the joint loss:

$$\mathcal{L} = \sum_{i=1}^{t} w_i \mathcal{L}_i, \tag{7}$$

where w_i is the task weight for task i. Stochastic gradient descent based methods can be used for the optimization and we use Adam [21] for M^{3}Rec.

4 EXPERIMENT

In this section, we move forward to evaluate the effectiveness of M^{3}Rec. We aim to answer the following questions:

RQ1 How does each task contribute to M^{3}Rec's performance?
RQ2 How does M^{3}Rec perform compared to the state-of-the-art methods in offline datasets?
RQ3 How does M^{3}Rec perform in online recommendation test?

4.1 Experimental Setup

We evaluate the performance of M^{3}Rec and other competitors in three online games on Tencent Games platform. The first game is LEGO®Cube and we have illustrated the details of each sequences and each tasks in Section 2. We also select two popular Tencent games belonging to the Role Playing Game (RPG) genre. Both games allow players to team up and combat with each other. Players can

Table 1: Dataset statistics.

Dataset	#sequence	#item/map	#user	#friend	#type
LEGO®Cube	1,884	2,446	20,319	18,435	7
T-game	60,811	62	75,518	14,707	12
Y-game	14,571	39	25,377	10,806	7

evolve their skills by purchasing items. We are required by our industry partners to anonymize the two games as Tgame and Ygame. For each player in Tgame, three behavior sequences are available: the *item sequence* each user has downloaded, the *type sequence* of each user's downloaded item, and the *friend sequence* each user has interacted with. It presents three recommendation problems: i.e., *next item prediction*, *next type prediction* and *next friend prediction*. For each player in Ygame, five behavior sequences are available: the *item sequence*, the *type sequence*, the *friend sequence*, the *evolve sequence* of items each user has utilized to evolve his/her skills, and the *buy sequence* of items each user has bought. It presents four recommendation problems: i.e., *next item prediction*, *next type prediction* and *next friend prediction*. It present five recommendation problems: i.e., *next item prediction*, *next type prediction*, *next friend prediction*, *next evolution prediction* for the next item to be evolved and *next purchase prediction* for the next item to be bought.

The datasets used in our offline test are samples from the three games. We do not sample users whose behavior sequences are missing. For example, users do not interact with any friend will not be sampled. We show the statistics of datasets in Table 1. Note that, in LEGO®Cube, players acquire maps and in the rest two games players acquire items. In each dataset, the number of users who are friended is calculated based on users (1) whose behavior sequences are not sampled, and (2) who appear in the friend sequence of a sampled user sequence.

4.2 Evaluation Metrics in Offline Test

For the offline tests in Section 4.3 and Section 4.4, we evaluate experimental results based on three evaluation metrics in the top-n recommendations, namely Hit Ratio (HR@n), Mean Reciptrocal Rank (MRR@n), and Normalized Discounted Cumulative Gain (NDCG@n). They are commonly adopted in evaluating recommendation performance [16, 32, 55, 66, 69].

4.3 Contributions of Multi-tasking (RQ1)

To understand how each task contribute to M^{3}Rec's performance (**RQ1**), we test three multi-task settings. We investigate the performance of modeling both main task *next map/item prediction* and one auxiliary tasks (i.e., *next type prediction* or *next friend prediction*), and the performance of modeling *all* three tasks. Note that here we only list three tasks that are common in all three games.

We compare different methods for HR@n, MRR@n, NDCG@n, where $n = 1, 2, \cdots, 10$. We determine the top three results out of eight outcomes based on each evaluation metric in each dataset. Then we score each multi-task setting by calculating the percentage of top-3 results: $s_i = \frac{n_i}{N}$, where n_i is the number of times that results generated by the i-th multi-task setting are the best three results

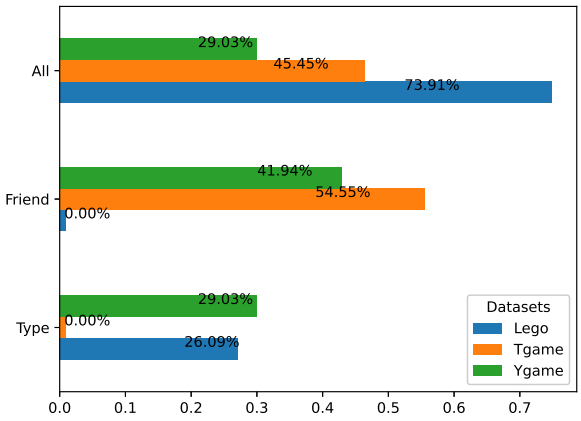


Figure 4: Percentage of top-3 results generated by different multi-tasking schemes.

based on any evaluation metric in any dataset. N is the overall number of top-3 results.

As shown in Figure 4, incorporating all auxiliary tasks generates the best results. It contributes to 29.03% of the top-3 results in Ygame, 45.45% of the top-3 results in Tgame and 73.91% of the top-3 results in LEGO®Cube. On the contrary, combining only one auxiliary task produces less stable results. For example, combining the main task with only next friend prediction does not perform well on LEGO®Cube. Its results are never among the top three best results in terms of any evaluation metric. Similarly, combining the main task with only next type prediction performs poorly on Tgame, with zero top three results in terms of any evaluation metric.

4.4 Offline Comparative Performance (RQ2)

To answer **RQ2**, we compare M^{3}Rec with the following methods as baselines in the offline test:

- GRU [55] is a state-of-the-art sequential recommendation model which applies Recurrent Neural Networks (RNN) with ranking-based loss functions. It improves [16] by data augmentation, and a method to account for shifts in the input data distribution.
- MARank [69] is recently proposed to unify both individual- and union-level item interaction into preference inference model from multiple views. It utilizes attention mechanism and residual neural network.
- STAMP [32] presents a novel short-term attention/memory priority model to capture users' general interests from the long-term memory of a session context, whilst takes into account users' current interests from the short-term memory of the last action.
- SR-GNN [66] models sequences as graph data and applies Gated Graph Neural Network [29] to extract complex transitions of items. An attention network is utilized to represent each sequence as a composition of global preference and current preference from the last action.

Table 2: Offline test on LEGO®Cube dataset with best results in bold.

Method	HR@1	HR@2	HR@5	HR@10	MRR@2	MRR@5	MRR@10	NDCG@2	NDCG@5	NDCG@10
GRU	0.0195	0.0357	0.0737	0.1356	0.0276	0.0378	0.0460	0.0357	0.0558	0.0770
MARank	0.0151	0.0329	0.0742	0.1311	0.0240	0.0350	0.0421	0.0329	0.0547	0.0737
STAMP	0.0184	0.0312	0.0725	0.1278	0.0248	0.0364	0.0434	0.0312	0.0539	0.0725
SR-GNN	0.0234	0.0379	0.0759	0.1088	0.0307	0.0402	0.0446	0.0379	0.0572	0.0684
GAT	0.0156	0.0335	0.0670	0.1328	0.0246	0.0334	0.0422	0.0335	0.0511	0.0737
M^3Rec	**0.0301**	**0.0513**	**0.0882**	**0.1423**	**0.0407**	**0.0505**	**0.0575**	**0.0513**	**0.0708**	**0.0891**

Table 3: Offline test on Tgame dataset with best results in bold.

	HR@1	HR@2	HR@5	HR@10	MRR@2	MRR@5	MRR@10	NDCG@2	NDCG@5	NDCG@10
GRU	0.5317	0.6827	0.8544	0.9445	0.6072	0.6546	0.6670	0.6827	0.7760	0.8075
MARank	0.5012	0.6737	0.8532	0.9417	0.5874	0.6369	0.6491	0.6737	0.7710	0.8020
STAMP	0.4798	0.6531	0.8378	0.9337	0.5664	0.6175	0.6307	0.6531	0.7535	0.7870
SR-GNN	0.5327	0.6901	0.8601	**0.9461**	0.6114	0.6587	0.6705	0.6901	0.7830	0.8130
GAT	0.4029	0.6094	0.8238	0.9309	0.5061	0.5663	0.5810	0.6094	0.7273	0.7648
M^3Rec	**0.5328**	**0.6910**	**0.8605**	**0.9461**	**0.6115**	**0.6589**	**0.6706**	**0.6910**	**0.7834**	**0.8133**

Table 4: Offline test on Ygame dataset with best results in bold.

	HR@1	HR@2	HR@5	HR@10	MRR@2	MRR@5	MRR@10	NDCG@2	NDCG@5	NDCG@10
GRU	0.5556	0.7266	0.9202	0.9882	0.6411	0.6943	0.7037	0.7266	0.8313	0.8552
MARank	0.5545	0.7230	0.9182	0.9863	0.6388	0.6926	0.7021	0.7230	0.8290	0.8528
STAMP	0.5200	0.6867	0.8931	0.9833	0.6034	0.6586	0.6714	0.6867	0.7962	0.8281
SR-GNN	0.5532	0.7305	0.9286	**0.9904**	0.6418	0.6966	0.7053	0.7305	0.8383	0.8602
GAT	0.4955	0.7031	0.9074	0.9822	0.5993	0.6575	0.6678	0.7031	0.8167	0.8430
M^3Rec	**0.5562**	**0.7380**	**0.9291**	0.9895	**0.6461**	**0.6987**	**0.7077**	**0.7380**	**0.8420**	**0.8634**

Table 5: Online test on LEGO®Cube

Method	NUV	CTR	ER	DR	CR
M^3Rec	1.00	100.00% ± 3.09%	100.00% ± 2.69%	100.00% ± 2.28%	100.00% ± 2.00%
IMF	0.12	67.32% ± 7.32%	70.29% ± 6.53%	70.44% ± 5.61%	71.41% ± 5.07%
SNS+IMF	0.12	58.05% ± 6.83%	66.58% ± 6.40%	70.06% ± 5.61%	67.79% ± 4.92%
RNN	1.31	92.20% ± 2.60%	96.16% ± 2.30%	96.48% ± 2.00%	95.08% ± 1.69%
HERec	1.10	73.01% ± 2.44%	75.29% ± 2.18%	72.53% ± 1.90%	72.71% ± 1.69%
oKNN	5.46	77.56% ± 1.14%	75.16% ± 1.02%	75.38% ± 0.86%	78.17% ± 0.77%
Random	0.60	65.53% ± 3.25%	44.56% ± 2.30%	53.61% ± 2.19%	58.42% ± 2.08%

Table 6: Online test on Tgame

Method	NUV	ARPU	CR(%)
M^3Rec	1.00	1.00 ± 0.02	100.00 ± 1.05
eALS	0.95	0.84 ± 0.02	93.98 ± 1.10
RF	0.94	0.92 ± 0.02	91.31 ± 1.12
IMF	1.30	0.79 ± 0.02	82.96 ± 0.97
Random	0.93	0.21 ± 0.01	32.20 ± 0.94

Table 7: Online test on Ygame

Method	NUV	ARPU	CR(%)
M^3Rec	1.00	1.00 ± 0.46	100.00 ± 20.62
IMF	5.19	0.60 ± 0.13	68.33 ± 7.50
POP	3.05	0.19 ± 0.07	41.60 ± 7.65

- GAT [41] collaboratively considers the sequence order and the latent order of user preference by constructing a session graph and utilizing a weighted attention graph layer.

All methods, including competitors and M^{3}Rec use an embedding size of 128, a learning rate of 0.0001 and a batch size of 128. For other special hyper-parameters of competitors, we use the recommended default settings given in their papers or open-source implementations.

The results of HR@n, MRR@n, and NDCG@n with $n = 1, 2, 5, 10$ on the three datasets are shown in Tables 2, 3 and 4. We omit MRR@1 and NDCG@1 as these two are identical with HR@1. For M^{3}Rec, the best performance multi-task setting is reported. We can observe that M^{3}Rec performs consistently best in terms of all evaluation metrics on all datasets. This shows that M^{3}Rec obtains very promising accuracy in the top-n sequential recommendations.

4.5 Online Comparative Performance (RQ3)

To validate that M^{3}Rec has indeed promoted recommendation performance (**RQ3**), we conduct an online evaluation through a A/B test. We randomly select a group of exposed users for each recommendation approach (e.g., the competitors or M^{3}Rec). Then we deliver top items with highest prediction score by each recommendation approach to the exposed users on the main task.

Competitors. Note that the competitors used in the online test are currently deployed in the online games. Due to various reasons such as security and efficiency, the recommender of each game might have differing models which are also different compared to those used in our offline test.

(1) **Competitors on LEGO®Cube**:

- IMF [19] is a latent factor model treating the data as positive and negative instances with vastly varying confidence levels.
- SNS+IMF improves IMF by leveraging the social network information. The confidence of each instance in the social network is computed by the efficient Personalized PageRank (PPR) algorithm on large graphs using the distributed computing framework [31].
- HERec [51] is a state-of-the-art recommendation model based on heterogeneous information network.
- RNN [16] deploys RNNs to predict next map.
- oKNN [5] is the online KNN algorithm, where each recommendation is made based on the majority voting from the user's cluster.
- Random is to randomly predict the next map.

(2) **Competitors on Tgame.**

- eALS [14] adopts an efficient element-wise Alternating Least Squares (eALS) learning technique in factorization machine [22] which weighs the missing data based on item popularity.
- RF [30] is the classic random forest prediction model.
- IMF [19] is the latent factor model which learns from weighted, positive and negative instances.
- Random is to randomly predict the next item.

(3) **Competitors on Ygame** include IMF as used in the other two games and POP which is to give the most popular item as the prediction for the next item.

Evaluation Metrics. The evaluation metrics are suggested by the operation team in Tencent game, which are designed for each game in the purpose of better operation. The following evaluation metrics are based on UV, which is the number of users exposed in each group:

(1) **Evaluation Metrics for LEGO®Cube**:

- Click Through Rate (CTR) is calculated as $CTR = C/UV$, where C is the total clicks received from the exposed users.
- Download Rate (DR) is calculated as $DR = D/UV$, where D is the number of exposed users who download the recommended map.
- Entrance Rate (ER) is calculated as $ER = E/UV$, where E is the number of exposed users who enter the recommended map in LEGO®Cube.
- Conversion Rate (CR) is calculated as $CR = PU/UV$, where PU is the number of purchasing users.

(2) **Evaluation metrics on Tgame and Ygame** include Conversion Rate (CR) as used in LEGO®Cube and Average Revenue Per User (ARPU) (i.e., $ARPU = R/UV$, where R is the total revenue and UV is the number of users exposed).

As an effort to quantifying the size of each experiment group without revealing the exact number of users exposed, we also give the Normalized User View (NUV). NUV is the number of users exposed in each experiment group divided by the number of users exposed in the group for M^{3}Rec. Thus the NUV of M^{3}Rec always equals one.

We report the results with the confidence level $rho \geq 0.95$ in Tables 5, 6 and 7. We can observe that M^{3}Rec significantly outperforms previously deployed competitors, in terms of all evaluation metrics that are designed by the operation team. This clearly suggests the great potential of M^{3}Rec in online gaming.

5 RELATED WORK

In this section, we will elaborate on the relevant works which can be concluded into three main paradigms: *General Recommender Systems*, *Sequential Recommender Systems* and *Multi-task Recommender Systems*.

5.1 General Recommender Systems

Recommender Systems (RS) have become an essential tool for solving information overload problem [46]. RS not only assists users in searching for desirable targets but also helps e-commerce platforms promote their products and boost sales [1]. Traditional RS do not consider sequential behaviors and they typically rely on collaborative filtering methods (CF), especially matrix factorization (MF) [22], to utilize historical user-item interactions for recommendation. MF factorizes the user-item interaction matrix into two low-dimensional latent matrices while preserving the inherent information from original user-item interactions. MF has proven its ability of modeling user preferences and item properties in Netflix Prize Challenge [3]. Due to its effectiveness when handling

large-scale data [23], MF has been successfully deployed in the industry (e.g., Facebook, Amazon and Netflix [1, 22]). One of the most challenging issues in traditional recommender systems is the cold-start problem, where historical data is not available for new users or items [1]. In order to alleviate the cold-start problem, many works have incorporated additional context information, which is also called auxiliary data or side information (e.g., social network [26, 27], review text [12, 60], image [59], structural data [9, 25], and location [34]), into recommendation models. However, traditional recommender systems only consider user-item interactions. It is difficult to use general recommenders for sequential recommendation task directly, since user sequential behaviors should also be modeled [28].

5.2 Sequential Recommender Systems

The research of sequential recommender systems (SRS) has emerged recently, as many real-world applications have sequence based traits [35, 63].

The pioneering works for sequential recommender systems (SRS) utilize Markov Chain (MC) which views the three-way interactions in SRS as two components: one is the interaction between the user and the next item and the other is the sequential history between previous items and the next item [8, 49, 50, 56, 71]. The former is well studied in Matrix Factorization (MF) [22] and the later can be sequentially modeled by MC. Therefore, MC based methods adopt the ideas from both MF and MC. The drawback of MC based model is that the state space quickly becomes unmanageable when trying to include all possible sequences of user selections [16]. Another line of works extends MC based SRS using the idea of "translation": the next item is viewed as the translation from previous item via the user [12, 13, 24]. The major advantage of these works is that they are much faster than other methods. There are other works using non-machine learning methods. For example, Migliorini et al. [37] deals with the production of sequences of recommendations for dynamic groups by considering the role of the context.

Hidasi et al. [16] firstly introduce Recurrent Neural Network (RNN) into sequential recommendation problem and proposed GRU4Rec which utilizes RNNs with a Gated Recurrent Unit (GRU) for SRS. Later, they extended GRU4Rec to exploit additional features (e.g., picture and text) by using parallel RNN architectures [17]. Due to the large performance gain of GRU4Rec over traditional methods, RNN based methods (including RNN, GRU and LSTM) have become prevalent in recent studies of SRS [6, 15, 20, 28, 33, 45, 52, 55, 58, 64, 65]. In addition to RNN, a few researchers explored the possibility of using other neural networks for SRS. Tuan and Phuong [57] harnessed 3D Convolutional Neural Network (CNN) and side information to enhance the accuracy of sequential recommendation. Wu et al. [66] investigated how to improve SRS with Graph Neural Network (GNN) and each sequence is then represented as the composition of the global preference and the current interest of that sequence using an attention network. Following Wu et al. [66], a few recent works explore the potential of GNN in SRS [41, 42, 68].

There are also some SRS considering the availability of user identity in each sequence. Due to the fact that the past and current user sequences can be simply concatenated to obtain longer sequence for the same user. Epure et al. [10] introduced the concept of medium-term behavior in addition to the existing short-term and long-term behaviors in personalized SRS for news and combined them together to enhance recommendation on the top of MC. Quadrana et al. [44] argued that concatenating user sequences when user identity is available will not yield the best result and proposed a Hierarchical RNN model with cross-session information transfer. Inspired by the success of language modeling, Hu et al. [18] modeled the information of the user and items in a sequence as the context and used probabilistic classifier to identify the item to be recommended next. Liu et al. [32] noticed the user interests drift in a long user sequence is not well modeled by conventional models. Therefore, they designed a short-term attention/memory priority model as a remedy. Song et al. [53] modeled social influence in sequential social recommender with RNN and a graph-attention neural network.

As explained in Section 1, conventional SRS are single-sequence and single-task and hence they are not suitable for the multi-sequence and multi-task recommendation task in Tencent Games platform.

5.3 Multi-task Recommender Systems

Multi-task learning has been successfully deployed in several applications [70]. Multi-task learning aim is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks.

As far as we know, multi-task learning has not been introduced to SRS before, but there are some efforts in using multi-task learning to improve other types of RS. Wang et al. [62] proposed OMTCF which models each user in online CF as an individual task. OMTCF not only update the weight vectors of the user (task) related to the current observed data, but also the weight vectors of some other users (tasks) according to a user interaction matrix. With the similar idea, Wang et al. [62] introduced a multi-task learning framework which learns multiple rating prediction models simultaneously (one for the active user and one for each of the related users). Chen et al. [7] designed a multi-task framework for music playlist recommendation, which can deal with three recommendation tasks (i.e., cold playlist recommendation, cold user recommendation and cold song recommendation) in parallel. Ni et al. [38] considered optimizing multiple search and recommendation tasks in e-commerce platform and learned universal user representations across multiple tasks for more effective personalization. Multi-task learning is also used in knowledge graph enhanced RS. For instance, knowledge graph completion task can be utilized to assist recommendation task [4, 61]. Moreover, the induction of explainable rules from knowledge graphs can be integrated with recommendation task. Ma et al. [36] and the two tasks complement each other in a multi-task framework.

6 CONCLUSION

In this paper, we study a new research problem which naturally arises from the recommendation scenario of online games. Using the ideas of GNN and multi-task learning, we propose a new method M^3Rec to fully utilize the complex information in the heterogeneous sequential recommendation scenario of online games.

Online evaluations in three games of Tencent Games platform illustrate the effectiveness of M^{3}Rec. In the future, we plan to enhance the interpretability of M^{3}Rec so that we can better understand the recommendation results.

ACKNOWLEDGMENTS

Chen Lin is supported by the Natural Science Foundation of China (No. 61972328) and Joint Innovation Research Program of Fujian Province China (No. 2020R0130). Hui Li is supported by the Natural Science Foundation of China (No. 62002303) and Natural Science Foundation of Fujian Province China (No. 2020J05001).

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook*. Springer.
[2] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *Proc. VLDB Endow.* 2, 1 (2009), 754–765.
[3] Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *SIGKDD Explorations* 9, 2 (2007), 75–79.
[4] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. 151–161.
[5] Olivier Cappé and Eric Moulines. 2007. Online EM Algorithm for Latent Data Models. *CoRR* abs/0712.4273 (2007). http://arxiv.org/abs/0712.4273
[6] Sotirios P. Chatzis, Panayiotis Christodoulou, and Andreas S. Andreou. 2017. Recurrent Latent Variable Networks for Session-Based Recommendation. In *DLRS@RecSys*. 38–45.
[7] Dawei Chen, Cheng Soon Ong, and Aditya Krishna Menon. 2019. Cold-start Playlist Recommendation with Multitask Learning. *arXiv Preprint* (2019). https://arxiv.org/abs/1901.06125
[8] Shuo Chen, Joshua L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *KDD*. 714–722.
[9] Danhao Ding, Hui Li, Zhipeng Huang, and Nikos Mamoulis. 2017. Efficient Fault-Tolerant Group Recommendation Using alpha-beta-core. In *CIKM*. 2047–2050.
[10] Elena Viorica Epure, Benjamin Kille, Jon Espen Ingvaldsen, Rébecca Deneckère, Camille Salinesi, and Sahin Albayrak. 2017. Recommending Personalized News in Short User Sessions. In *RecSys*. 121–129.
[11] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2019. Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. *arXiv Preprint* (2019). https://arxiv.org/abs/1905.01997
[12] Alberto García-Durán, Roberto Gonzalez, Daniel Oñoro-Rubio, Mathias Niepert, and Hui Li. 2020. TransRev: Modeling Reviews as Translations from Users to Items. In *ECIR*, Vol. 12035. 234–248.
[13] Ruining He, Wang-Cheng Kang, and Julian J. McAuley. 2017. Translation-based Recommendation. In *RecSys*. 161–169.
[14] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*. 549–558.
[15] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *CIKM*. 843–852.
[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.
[17] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys*. 241–248.
[18] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. 2017. Diversifying Personalized Recommendation with User-session Context. In *IJCAI*. 1858–1864.
[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.
[20] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys*. 306–310.
[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[22] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.
[23] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: Fast and Exact Inner Product Retrieval in Recommender Systems. In *SIGMOD*. 835–850.
[24] Hui Li, Ye Liu, Nikos Mamoulis, and David S. Rosenblum. 2020. Translation-Based Sequential Recommendation for Complex Users on Sparse Data. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1639–1651.
[25] Hui Li, Yu Liu, Yuqiu Qian, Nikos Mamoulis, Wenting Tu, and David W. Cheung. 2019. HHMF: hidden hierarchical matrix factorization for recommender systems. *Data Min. Knowl. Discov.* 33, 6 (2019), 1548–1582.
[26] Hui Li, Dingming Wu, and Nikos Mamoulis. 2014. A revisit to social network-based recommender systems. In *SIGIR*. 1239–1242.
[27] Hui Li, Dingming Wu, Wenbin Tang, and Nikos Mamoulis. 2015. Overlapping Community Regularization for Rating Prediction in Social Recommender Systems. In *RecSys*. 27–34.
[28] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. 1419–1428.
[29] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.
[30] Andy Liaw and Matthew Wiener. 2002. Classification and Regression by randomForest. *R News* 2, 3 (2002), 18–22. https://CRAN.R-project.org/doc/Rnews/
[31] Wenqing Lin. 2019. Distributed Algorithms for Fully Personalized PageRank on Large Graphs. In *WWW*. 1084–1094.
[32] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.
[33] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling User Session and Intent with an Attention-based Encoder-Decoder Architecture. In *RecSys*. 147–151.
[34] Ziyu Lu, Hui Li, Nikos Mamoulis, and David W. Cheung. 2017. HBGG: a Hierarchical Bayesian Geographical Model for Group Recommendation. In *SDM*. 372–380.
[35] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Model. User-Adapt. Interact.* 28, 4-5 (2018), 331–390.
[36] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *WWW*. 1210–1221.
[37] Sara Migliorini, Elisa Quintarelli, Damiano Carra, and Alberto Belussi. 2019. Sequences of Recommendations for Dynamic Groups: What Is the Role of Context?. In *BigData Congress*. 121–128.
[38] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *KDD*. 596–605.
[39] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. 2012. Fast Group Recommendations by Applying User Clustering. In *ER*, Vol. 7532. 126–140.
[40] Auste Piliponyte, Francesco Ricci, and Julian Koschwitz. 2013. Sequential Music Recommendations for Groups by Balancing User Satisfaction. In *UMAP Workshops*, Vol. 997.
[41] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *CIKM*. 579–588.
[42] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation. In *SIGIR*. 669–678.
[43] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.
[44] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*. 130–137.
[45] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-Based Recommendation. In *AAAI*. 4806–4813.
[46] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. *Recommender Systems Handbook*. Springer.
[47] Senjuti Basu Roy, Laks V. S. Lakshmanan, and Rui Liu. 2015. From Group Recommendations to Group Formation. In *SIGMOD*. 1603–1616.
[48] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.
[49] Guy Shani, Ronen I. Brafman, and David Heckerman. 2002. An MDP-based Recommender System. In *UAI*. 453–460.
[50] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *Journal of Machine Learning Research* 6 (2005), 1265–1295.
[51] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Trans. Knowl. Data Eng.* 31, 2 (2019), 357–370.
[52] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *DLRS@RecSys*. 2–9.
[53] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM*. 555–563.
[54] Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. 2020. Fair sequential group recommendations. In *SAC*. 1443–1452.

[55] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS@RecSys*. 17–22.
[56] Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for detecting topics of user sessions. In *RecSys*. 33–40.
[57] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *RecSys*. 138–146.
[58] Bartlomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *RecSys*. 273–276.
[59] Cheng Wang, Mathias Niepert, and Hui Li. 2018. LRMM: Learning to Recommend with Missing Modalities. In *EMNLP*. 3360–3370.
[60] Cheng Wang, Mathias Niepert, and Hui Li. 2020. RecSys-DAN: Discriminative Adversarial Networks for Cross-Domain Recommender Systems. *IEEE Trans. Neural Networks Learn. Syst.* 31, 8 (2020), 2731–2740.
[61] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*.
[62] Jialei Wang, Steven C. H. Hoi, Peilin Zhao, and Zhiyong Liu. 2013. Online multi-task collaborative filtering for on-the-fly recommender systems. In *RecSys*. 237–244.
[63] Shoujin Wang, Longbing Cao, and Yan Wang. 2029. A Survey on Session-based Recommender Systems. *arXiv Preprint* (2029). https://arxiv.org/abs/1902.04864
[64] Zhitao Wang, Chengyao Chen, Ke Zhang, Yu Lei, and Wenjie Li. 2018. Variational Recurrent Model for Session-based Recommendation. In *CIKM*. 1839–1842.
[65] Chen Wu and Ming Yan. 2017. Session-aware Information Embedding for E-commerce Product Recommendation. In *CIKM*. 2379–2382.
[66] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*. 346–353.
[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Knowl. Data Eng.* (2020).
[68] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*. 3940–3946.
[69] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-Order Attentive Ranking Model for Sequential Recommendation. In *AAAI*. 5709–5716.
[70] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. *arXiv Preprint* (2017). https://arxiv.org/abs/1707.08114
[71] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using Temporal Data for Making Recommendations. In *UAI*. 580–588.