Category-aware Collaborative Sequential Recommendation

Renqin Cai[1], Jibang Wu[1], Aidan San[1], Chong Wang[2], Hongning Wang[1]
[1]University of Virginia, Charlottesville, VA, USA
[2]Bytedance, Bellevue, WA, USA
[1]{rc7ne, jw7jb, aws9xm, hw5x}@virginia.edu, [2]chong.wang@bytedance.com

ABSTRACT

Sequential recommendation is the task of predicting the next items for users based on their interaction history. Modeling the dependence of the next action on the past actions accurately is crucial to this problem. Moreover, sequential recommendation often faces serious sparsity of item-to-item transitions in a user's action sequence, which limits the practical utility of such solutions.

To tackle these challenges, we propose a Category-aware Collaborative Sequential Recommender. Our preliminary statistical tests demonstrate that the in-category item-to-item transitions are often much stronger indicators of the next items than the general item-to-item transitions observed in the original sequence. Our method makes use of item category in two ways. First, the recommender utilizes item category to organize a user's own actions to enhance dependency modeling based on her own past actions. It utilizes self-attention to capture in-category transition patterns, and determines which of the in-category transition patterns to consider based on the categories of recent actions. Second, the recommender utilizes the item category to retrieve users with similar in-category preferences to enhance collaborative learning across users, and thus conquer sparsity. It utilizes attention to incorporate in-category transition patterns from the retrieved users for the target user. Extensive experiments on two large datasets prove the effectiveness of our solution against an extensive list of state-of-the-art sequential recommendation models.

CCS CONCEPTS

• **Information systems → Personalization**; **Recommender systems**; • **Computing methodologies → Neural networks**.

KEYWORDS

sequential recommendation, contextualized recommendation, collaborative learning, neural networks

ACM Reference Format:
Renqin Cai, Jibang Wu, Aidan San, Chong Wang, Hongning Wang. 2021. Category-aware Collaborative Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462832

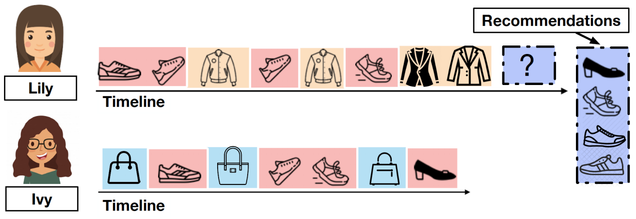

Figure 1: An illustration of collaborative sequential recommendation. Each user's actions are indexed chronologically. The recommender system needs to predict which items to recommend to the user Lily based on her and another user Ivy's past actions.

1 INTRODUCTION

The essence of sequential recommendation is to model the influence from past actions on the next action. This has been considered as a sequential prediction problem, and various sequence models borrowed from other fields have been explored [3, 8, 9, 17, 24, 32, 35]. From the earliest Markov models [24] to recent neural sequence models, such as Recurrent Neural Network (RNN) or self-attention [8, 30], models with a stronger capacity in capturing complex and high-order dependence among actions have shown to achieve better recommendation quality.

However, most of existing solutions treat a user's action history as a long sequence [2, 5, 8]. Such simplification usually ignores the fine-grained dependency structure in the action sequence. Consider the example illustrated in Figure 1. For the user Lily, the reoccurred transitions from clothes to shoes suggest her next action is very likely to be related to shoes, and the series of her previously browsed shoes suggest her general preference on sports shoes. But her recently browsed business suits suggest her current intent in formal outfits. As a result, it is no longer appropriate for the system to follow her general preference to recommend sports shoes; instead, recommending formal shoes becomes a better choice. Such observation informs us that sequential recommendation should be context-aware: under different contexts, the prediction of next action should depend on different subsequences of past actions.

Another challenge in sequential recommendation is data sparsity. Observations about individual user's actions are known to be sparse [12], not to mention the transitional patterns that could be covered in a single user's action sequence. For example, in the example shown in Figure 1, the user Lily has never visited any formal shoes in the past. Hence, even with the knowledge that formal shoes should be recommended next, it is still clueless for the system to predict which specific type/brand to start with.

As a remedy to the data sparsity issue, collaborative learning methods have been recently introduced to sequential recommendation [14, 15, 19, 25]. The basic idea is to exploit users with similar past action sequences for the next action prediction. As a typical solution of this type, Wang et al. [34] modeled users' action sequences with RNNs and retrieved neighboring users based on the user latent states learnt by RNNs. Then the target user's representation is combined with the retrieved neighboring users' representations for the next item prediction. However, user similarity is still measured by the entire sequence of past actions in this type of solutions. As we argued before, neglecting the context in sequential recommendation introduces inaccurate dependency on the past actions, and therefore erroneous neighborhood for next action prediction. Consider the example in Figure 1 again. When looking at their entire action sequences, Lily and Ivy might not be considered as neighbors, as Lily visited mostly shoes and clothes while Ivy visited mostly shoes and handbags. But the subsequences of shoes browsed by these two users make them closer. Especially the transitions from sport shoes to formal shoes in Ivy's subsequence will be very helpful in predicting Lily's next action about formal shoes. Therefore, the neighborhood modeling in sequential recommendation should also be context-aware.

Nevertheless, the context, under which a user takes the next action, is not observable by the system [1]. We have to look for proxies of it. We believe a good proxy of context should: 1) be widely available in sequential recommendation problems, and 2) enhance the modeling of dependence on past actions. In this work, we consider two types of proxies: a user's most recently interacted items before the next action, and the category of the next item. Treating the recent items as the context of the next action has been a common practice in many existing studies [20, 33, 36]. Moreover, we consider item category, a type of widely available metadata about items, also provides context information. Our statistical tests on two large public recommendation datasets prove the transitional patterns among actions in the category-specific subsequences are significantly stronger than those in the original action sequences without considering the categories. We defer the details of our statistical tests to Section 3 and the description of the dataset to Section 4. To differentiate these two types of contexts, we term the context inferred from recent items as *episodic context* and that inferred from the item category as *category context.* These two types of proxies provide complementary views about the context under which the user is taking the next action [4, 22]. For instance, in Figure 1, the episodic context suggests Lily is looking for formal outfits, an intent shared by actions in close proximity, while the category context suggests Lily is looking for shoes, an intent specific to the next item.

Based on our insight discussed above, we propose a CategOry-aware COllaborative sequential Recommender (CoCoRec), which draws dependence of the next action on historical actions based on the target user's action sequence, item categories and neighboring users' action sequences. CoCoRec is composed of three key components: an in-category encoder, a context encoder, and a collaboration module. The in-category encoder utilizes self-attention to model item transition patterns in category-specific action subsequences. The context encoder infers the episodic context and the category context for the next action prediction. First, it uses self-attention to model the user's most recent actions to obtain the episodic context. Second, it uses recent actions' categories to predict the category of the next action, and then based on this prediction, it uses a gating network to activate the corresponding in-category item-to-item transitions. The collaboration module uses a memory tensor to record users' in-category preferences. For each target user, the collaboration module retrieves neighboring users with similar in-category preferences. Combining the episodic context, the in-category preferences of target user, and the neighbors' in-category preferences, CoCoRec predicts the next item for the target user.

To investigate the effectiveness of CoCoRec for sequential recommendation, we performed extensive experiments on two large public recommendation datasets. Compared with a list of state-of-the-art solutions for sequential recommendation, CoCoRec improved the recommendation quality in both recall and MRR. Our ablation analysis further demonstrated the importance of modeling the in-category user preferences and collaborative learning among users with similar in-category preferences for CoCoRec to achieve high quality recommendation performance.

2 RELATED WORK

The improvement of sequence models' capacities in capturing complex and high-order sequential patterns has unleashed the development in sequential recommendation. Fixed- and varying-order Markov models are among the earliest attempts [3, 24], which assume the prediction only depends on the recent several actions. Factorizing Personalized Markov Chains [24] is a typical model of this kind, which combines factorization machine [23] with a Markov model. It improves over vanilla matrix factorization by introducing sequential order among historical actions into factorization. However, the Markovian assumption also limits the performance of such models: as the state-space grows exponentially with respect to the order of dependence, this type of solution can hardly capture high order dependence in practice.

Neural sequence models, such as RNN, Long Short-Term Memory (LSTM) [9], Gated Recurrent Unit (GRU) [6] and self-attention [30] models, have been adopted to address the limitations in Markov models [5, 8, 9, 17, 32, 35]. For example, Hidasi et al. [8] applied RNNs to predict the next action based on actions in a session of which the boundary is defined in regard to the idle duration between two consecutive actions. Li et al. [17] used attention to model influence from the most related actions in the session. SASRec [13] and BERT4Rec [27] extended the scope of self-attention models to sequential recommendations.

Besides vanilla application of existing neural sequence models, problem-specific customizations are proposed to enhance the modeling of action sequences. Tang et al. [28] has developed a solution composed of a mixture of sequence models to capture both long-term and short-term action dependence. Hierarchical neural network models have been applied to model users' sequential preferences across different sessions [21, 37, 39].

Nevertheless, Wu et al. [36] found that the session assumption could be the bottleneck of these models, as the influence from past actions does not necessarily differ with respect to the manually defined session boundaries. In contrast, our fine-grained action dependency modeling is supported by statistical tests, i.e., segmenting action sequences into sub-sequences with respect to item categories enhances sequential dependency modeling.

Collaborative learning has been introduced to sequential recommendation to address the data sparsity issue. Based on the social network among users, Song et al. [26] combined the transition patterns of neighbors in the social network into the next item prediction of the target user. Lifting the requirements of pre-existing social networks, Jannach and Ludewig [12] measured user relatedness by the degree of item overlap among action sequences. Actions from the k-nearest neighbors are introduced to the target user's prediction. As a follow-up, Wang et al. [34] measured user similarities based on the latent states learnt for users by RNNs; and directly combined neighboring users' latent states with target user's latent state for the next item prediction. To reduce the search space of neighbors, Pan et al. [19] made the initial next item predictions based on the target user's action sequence, and then utilized the initial item predictions to filter irrelevant users. All the aforementioned methods used entire action sequences to measure user similarity. However, under different context, the importance of past actions in representing users is different. Failing to characterize target user's ongoing context when retrieving neighbors prevents collaborative learning from helping the next item prediction of the target user.

Another line of work on sequential recommendation utilized item categories as proxies of action context and incorporated the context into various models to improve the modeling of action sequences. Treating the prediction of the next item category as an extra task along with the prediction of the next item, a multi-task learning based solution has been developed to predict both the next item and its category [38]. Treating the item category as a condition, a generative adversarial network based solution validates the next item prediction based on the category of the predicted item [22]. However, the in-category sequential transition patterns are ignored by these solutions.

Besides item category, other types of external knowledge about items have also been utilized for sequential recommendation, like knowledge bases [10, 31] and category taxonomies [7, 11]. The relations among items defined in knowledge base are utilized to capture the dependence among actions [10, 31]. Huang et al. [11] incorporated multi-hop categories to memory network to structure the dependency. Likewise, Gu et al. [7] proposed to hierarchically model actions based on the hierarchy of item category. However, the limited availability of knowledge bases or category taxonomies in different recommendation scenarios directly restricts the application of these solutions in practice.

3 METHOD

We study sequential recommendations for a set of users $u \in U$ over a set of items $v \in V$ from a set of item categories $c \in C$. We denote an action as a tuple $a_i = (v_i, c_i)$, where i is the index of the action in a sequence, v_i is the item that the user interacts with and c_i is the item's category. Different actions may be associated with the same item and each item is associated with a unique category. A sequence of N actions from user u is denoted as $S_u = \{a_1, a_2, ..., a_N\}$, which is ordered chronologically with respect to the timestamps of actions. A subsequence of actions under category c is denoted as $S_u^c = \{a_1^c, ..., a_T^c\}$ where T represents the number of actions in this subsequence and actions are still ordered chronologically. Given S_u from user u, the goal of sequential recommendation is to rank

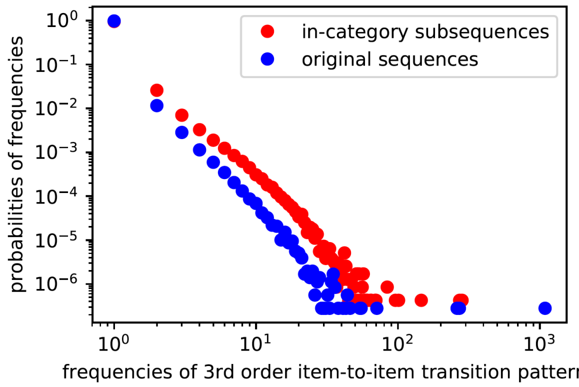

Figure 2: Result of the statistical dependence analysis on Taobao dataset. The distribution of frequencies of 3rd-order item-to-item transition patterns in in-category subsequences are as the red points show. The distribution of the frequencies of 3rd-order item-to-item transition patterns in the original sequences are as the blue points show.

items for this user to consider as the next item v_{N+1} in the next action a_{N+1}. When no ambiguity is incurred, we omit the subscript u to simplify our subsequent discussions.

3.1 Data-Driven Statistical Analyses

Before introducing our proposed solution, we first describe our statistical analyses about users' sequential behaviors on two public recommendation datasets: Taobao dataset and BeerAdvocate dataset. The number of total actions on both datasets are larger than 500K, which ensures the statistical significance of our analyses. The findings in these analyses directly lead to the design of our solution. More details about these datasets are in Section 4.

We investigate the dependence structure introduced by item categories. We segment a sequence of actions into multiple subsequences, where each subsequence consists of actions of the same item category. We count the frequency of Mth-order item-to-item transition patterns within subsequences and original sequences respectively. Specifically, the Mth-order item-to-item transition pattern refers to M items appearing consecutively in a given sequence (or a subsequence), e.g., $\{v_i, ..., v_{i+M-2}, v_{i+M-1}\}$. Figure 2 (a) shows on Taobao dataset, the probability of the 3rd-order item transition patterns appearing multiple times within subsequences is significantly higher than that in the original sequences. Due to space limit, we omit the results on BeerAdvocate dataset, where we obtained similar observations. By varying M from 2 to 10, we observed similar results. These findings strongly support our decision of using item category to structure actions to enhance the modeling of the action dependence.

3.2 Category-aware collaboration Sequential Recommender

Propelled by the findings in our statistical analyses, we propose a CategOry-aware COllaborative sequential Recommender (CoCoRec). In a nutshell, CoCoRec is composed of three modules: an in-category encoder, a context encoder, and a collaboration module. First, to model user preferences under a category, we segment an

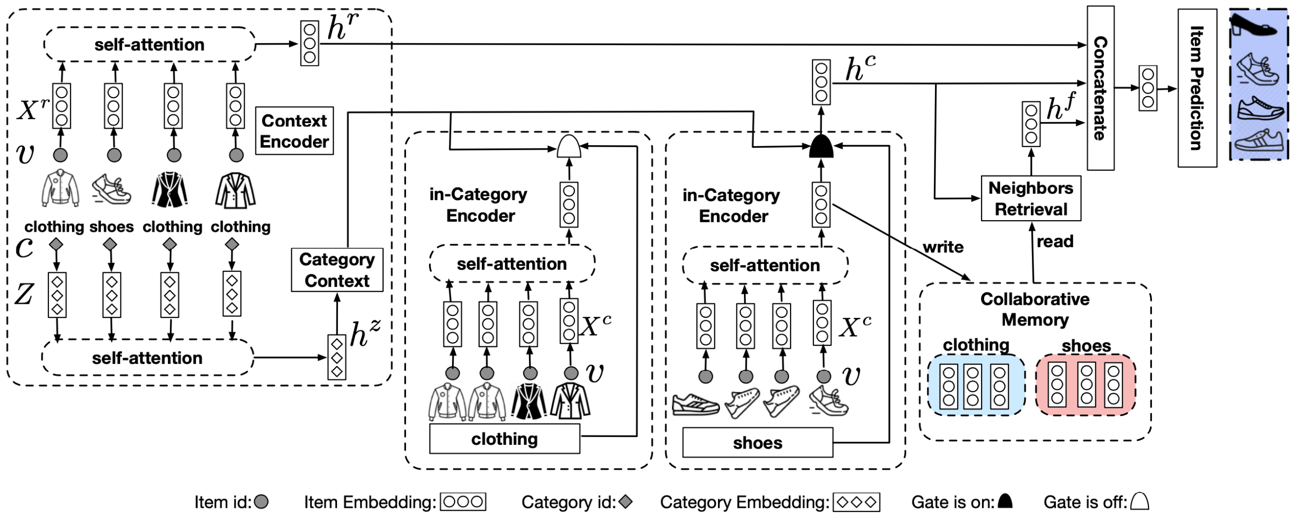

Figure 3: Overview of CoCoRec. In CoCoRec, an action sequence is decomposed into multiple subsequences with respect to the item category associated with each action. The in-category encoder encodes the category-specific action subsequences into latent vectors representing users' in-category preferences. The context encoder predicts the category of the next action to activate the corresponding in-category item-to-item transitions for the next item prediction. The context encoder infers the episodic context of the next action based on recent items. To address the sparsity issue, the collaboration module retrieves neighbors based on users' encoded in-category preferences. Based on signals from these three sources, CoCoRec predicts the next item and make recommendations to the user.

action sequence into multiple subsequences with respect to item categories and each subsequence is restricted to contain actions of the same item category. The in-category encoder utilizes self-attention to model in-category item-to-item transition patterns in the subsequences. In order to determine which in-category preference to use for the next item prediction, the context encoder predicts the next category based on the categories of recent actions with self-attention. Second, to model the episodic context, the context encoder utilizes another self-attention to model the item-to-item transition patterns among recent actions. Third, to leverage the neighboring users' in-category item-to-item transition patterns, we retrieve users with similar in-category preferences with regard to the target user's in-category preferences, based on the context encoder's next category prediction. Finally, the next item prediction is made based on the episodic context, the in-category user preferences, and neighboring users' in-category preferences.

In the following, we dive into the details of each component of CoCoRec to discuss about their designs.

3.2.1 In-category Encoder. The in-category encoder is designed to obtain the category-specific user preferences. To capture high-order item-item transition patterns, we choose a self-attention network for the in-category encoder. The self-attention network parameters are shared across categories to reduce model complexity, i.e., multi-task learning via parameter sharing.

For each of the $|C|$ categories, the in-category encoder learns a hidden representation of the user preferences respectively. Without loss of generality, we take the encoding process for an action subsequence of category c as an example to illustrate our design details. The item subsequence $[v_1^c, \ldots, v_T^c]$, which are associated with the action subsequence, are projected through the input item embedding layer $E_{in} \in \mathbb{R}^{|V| \times d_{in}}$ into a set of dense vectors $X^c = [e_{v_1^c}, \ldots, e_{v_T^c}]$ where $X^c \in \mathbb{R}^{T \times d_{in}}$. The relative positions $[T, \ldots, 1]$ of these actions to the next action are projected through the position embedding layer $P \in \mathbb{R}^{T \times d_{in}}$ into $P^c = [P_T^c, \ldots, P_1^c]$. Taking the dense vectors $X^c + P^c$ as input, the self-attention network outputs the representation of the user preferences in this category, i.e., $h^c \in \mathbb{R}^{d_h^c}$ by $h^c = \text{self-attention}(X^c + P^c)$.

The self-attention network is composed of n_l layers of a multi-head attention block and a point-wise feed-forward network block [18, 30]. Due to the recursive nature of these multiple layers of blocks, we use jth layer to explain the mechanism. There are n_h heads in a multi-head attention block with d_a hidden units. For the ith attention head, the attention block transforms the input latent states $H^j \in \mathbb{R}^{T \times d_a}$ of an action sequence into the output states as,

$$A_i^j = \text{Attention}(H^j W_i^Q, H^j W_i^K, H^j W_i^V)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_a / n_h}}\right)$$

where the projection matrices $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_a \times d_a / n_h}$ are learnable parameters mapping H^j into query Q, key K and value V representations. In addition, $\sqrt{d_a / n_h}$ is the scaling factor to encourage a softer attention distribution for avoiding extremely small gradients.

We then concatenate the output states obtained by all heads and project the concatenated representation to $A^j = [A_1^j, \ldots, A_{n_h}^j] W^O$, where $W^O \in \mathbb{R}^{d_a \times d_a}$ is another projection matrice. Feeding this representation into the a fully connected feed-forward network (FFN^j) and then a layer normalization network (LN), we obtain self-attended hidden vectors of this sequence as:

$$H^{j+1} = \text{LN}\left(H^j + FFN^j(A^j)\right)$$

In the in-category encoder, the input to the self-attention network is $H^0 = [e_{v_1^c}+P_T^c, \ldots, e_{v_T^c}+P_1^c]$ and $d_h^c = d_a$. The self-attention network transforms H^0 into $H^{n_l} = [h_1^c, \ldots, h_T^c]$ and uses the hidden state of the last action as the output of the self-attention network to represent the user preferences within category c, i.e., $h^c = h_T^c$.

3.2.2 Context Encoder. The context encoder is to obtain the context (both category and episodic context) for the next item prediction. To decide which in-category user preferences to leverage, we predict the category of the next item. Specifically, we use a top-k gating network to obtain the category context. In addition, we use a recency encoder to capture the episodic context buried in recent items.

• ***Top-k gating network.*** We use the categories of recent items as the input to the top-k gating network. The categories of the most recent L items are projected through the input category embedding layer $E_{in}^z \in \mathbb{R}^{|C|\times d_{in}^z}$ into a set of dense vectors, $Z = [e_{c_{N-L}}^z, \ldots, e_{c_N}^z]$. The relative positions of recent actions, denoted as $[L, \ldots, 1]$, are projected through the position embedding layer $P_{cate} \in \mathbb{R}^{L\times d_{in}^z}$ into a set of dense vectors $P^z = [P_L^z, \ldots, P_1^z]$.

In the top-k gating network, a self-attention network transforms the dense category vectors $Z+P^z$ into hidden representations and utilizes the hidden representation at the last action as the representation summarizing the category information of recent actions, as $h^z = \text{self-attention}(Z+P^z) \in \mathbb{R}^{d_h^z}$. Feeding h^z into the output category embedding layer E_{out}^z and then a softmax layer, the top-k gating network generates a probability distribution over all categories:

$$p(\hat{c}_{N+1} = j) \propto \exp(\langle h^z, e_j^z \rangle), \tag{1}$$

where $p(\hat{c}_{N+1} = j)$ denotes the probability of category j being the category of the next item and e_j^z is the category embedding of the category j. To account for the uncertainty in next category prediction, the gating network selects top-k most probable categories according to Eq. (2):

$$\{c_j\}_{j=1}^k = \underset{j'}{\arg\text{topk}} \left(\{p(\hat{c}_{N+1} = j')\}_{j'=1}^{|C|}\right) \text{ where } c_j \in C. \tag{2}$$

With respect to these k predicted categories, we include corresponding in-category user preferences to predict the next item, i.e., the hidden representations $\{h^{c_j}\}_{j=1}^k$ from the in-category encoder are selected. Besides, we also count the probabilities $p(\hat{c}_{N+1})$ of these k categories in the prediction of the next item, which we will discuss later. Because of this design, this top-k gating network is still differentiable, i.e., the training loss can be propagated back to update the category embeddings.

• ***Recency encoder.*** The recency encoder is introduced to infer the episodic context from recent actions. Due to their close proximity to the next action, the recent actions reflect the ongoing intent of the next action. For example, in Figure 1, Lily's recent actions suggest she is looking for formal outfits. To capture high-order dependence, we adopt another self-attention network to encode item-to-item transition patterns among recent actions.

Specifically, the input of the recency encoder is the most recent L items in the original sequence S, i.e., $[v_{N-L}, \ldots, v_N]$, which are projected through the input item embedding layer E_{in} into dense vectors $X^r = [e_{v_{N-L}}, \ldots, e_{v_N}]$, where $X^r \in \mathbb{R}^{L\times d_{in}}$. The relative positions of recent actions, denoted as $[L, \ldots, 1]$, are projected through the position embedding layer $P_{recent} \in \mathbb{R}^{L\times d_{in}}$ into a set of dense vectors $P^r = [P_L^r, \ldots, P_1^r]$. Then the self-attention network transforms vectors X^r+P^r into hidden states and we use the hidden state of the last action as the representation of the inferred episodic context as $h^r = \text{self-attention}(X^r + P^r) \in \mathbb{R}^{d_h^r}$. This episodic context is utilized for the next item prediction. We should note the recency encoder examines the recent items disregarding their categories, as the episodic context refers to information shared by actions with close proximity beyond specific categories. It provides complementary view of the next action besides the category context.

3.2.3 Collaboration Module. The collaboration module is designed to leverage neighboring users' in-category preferences for the target user's next item prediction. Due to the sparsity of observations in individual users' actions, the item-to-item transition patterns in a single user are expected to be sparse. Collaborative learning across neighboring users with similar preferences has the potential to mitigate the sparsity issue. Because our statistical analyses demonstrate that the in-category item-to-item transition patterns strongly suggest the user's preferences of the next item, we utilize the in-category subsequences to obtain the similarities among users. Then we combine the neighboring users' information based on their similarities for the next item prediction.

The collaboration module uses a memory tensor $Mem \in \mathbb{R}^{|C|\times F\times d_h^c}$ to record users' in-category preferences, denoted as "collaborative memory" in Figure 3. Specifically, it records the latent states of last F users for each category c in a chronological order.

• ***Reading operation.*** Given the target user's in-category preferences h_u^c of the category c, we compute its similarities to F latent states of user preferences of this category c in Mem: $\text{sim}(h_u^c, h_i^c) \propto \exp(\langle h_u^c, h_i^c \rangle)$. We choose the top-$f$ similar users as the neighbors and take a weighted sum of their representations by the corresponding similarities as the neighborhood representation for the next item prediction, as $h^f = \sum_{i'}^{\text{top-f}} \text{sim}(h_u^c, h_{i'}^c) h_{i'}^c$.

• ***Writing operation.*** We randomly initialize the memory tensor Mem and update it with the latest user's in-category preference representations h^c which are outputs of the in-category encoder. The memory tensor is organized as a queue: the collaboration module pushes the most recently served user's representations of the category c to the memory tensor, while popping out the representations of users inactive for a long time.

3.2.4 Next Item Prediction. Based on in-category user preferences of the predicted top k categories, neighboring users' in-category preferences, and episodic context inferred from the most recent items, CoCoRec predicts the next item. Specifically, we concatenate these three representations for each of k categories and project concatenated representations into the output item embedding space with a feed-forward network layer (FFN). The mixture of obtained representations is considered as the user representation for predicting the next item as $h_j = \text{FFN}(h^r \oplus h^{c_j} \oplus h^{f_j})$, where $j = \{1, ..., k\}$.

Matching the user representation with the item embeddings E_{out}, we obtain the ranking scores of items by,

$$\text{score}(\hat{v}_{N+1}) = \sum_{j=1}^{k} \text{score}_j(\hat{v}_{N+1}) p(\hat{c}_{N+1} = c_j) \tag{3}$$

where $\text{score}_j(\hat{v}_{N+1}) = \text{softmax}(\langle h_j, E_{out}\rangle)$. CoCoRec ranks the items with respect to their predicted scores $score(\hat{v}_{N+1})$ in a descending order as recommendations to the user.

3.3 Model Training & Inference

We train CoCoRec in an end-to-end fashion by minimizing the loss on the predictions of both the next item and its category, where the cross entropy loss is adopted.

Since the item space can be very large in practice, we apply the negative sampling trick to compute the loss of item predictions. For each positive item, we randomly sample N_s items as negative instances according to their popularities in training dataset. Thus, the loss of item prediction is,

$$L_{item} = -\sum_{v=1}^{N_s+1} \delta(v_{N+1}=v) \log p(\hat{v}_{N+1} = v)$$

$$p(\hat{v}_{N+1} = v) = \text{softmax}\Big(score(\hat{v}_{N+1} = v)\Big)$$

where $\delta(\cdot)$ is an indicator function, v_{N+1} is the ground-truth item, and $\hat{v}_{N+1}$ is the model's prediction. Likewise, we compute the loss against all categories,

$$L_{cate} = -\sum_{j=1}^{C} \delta(c_{N+1} = j) \log p(\hat{c}_{N+1} = j).$$

where the $p(\hat{c}_{N+1} = j)$ is computed by Eq (1). The joint loss is thus computed as

$$L = \lambda \times L_{item} + (1 - \lambda) \times L_{cate}.$$

where λ is a hyper-parameter controlling the weight of these two losses in the objective function.

In addition, we modify the training scheme to deal with the discrepancy between training stage and testing stage. During training, the ground-truth category of the next item is available. Thus, we can directly choose the in-category user preferences of the ground-truth category for the next item prediction. In contrast, during testing, the next category is unknown. The errors of the next category prediction will be propagated to the next item prediction. To mitigate this issue, we separate the training phase into two stages. In the early stage of training the model, we directly use the ground-truth category for the next item prediction. In the second stage, we use the top-k predicted categories. Particularly, we start the second stage model training only when the accuracy of category prediction stops increasing.

4 EXPERIMENTS

In this section, we study the effectiveness of CoCoRec for sequential recommendation. We first describe two evaluation datasets, followed by the implementation details of our model on these two datasets. Then we compare CoCoRec against an extensive set of baselines, ranging from heuristic solutions to state-of-the-art sequential recommendation solutions. In addition, a complete ablation analyses illustrates the importance of modeling the in-category user preferences and collaborative learning among users with similar in-category preferences. We also study the influence of the hyper-parameters on the performance of CoCoRec.

4.1 Datasets

We performed the evaluation on Taobao dataset [1] and BeerAdvocate dataset [2], which are both publicly available. The Taobao dataset contains sequences of user actions from the online shopping website taobao.com. Each action is associated with a user ID, an item ID, a category ID of the item, and a timestamp of the action. Due to privacy concerns, the semantic meanings of categories are not available. We randomly sampled 100,000 sequences from November 25, 2017 to December 3, 2017 for our experiments, where we used the actions in the first 7 days as the training set, actions on the 8th day as the validation set, and actions on the 9th day as the test set. We removed items associated with fewer than 20 actions, and removed users with fewer than 20 or more than 300 actions. We merged categories which have fewer than 100 items into a special category, denoted as category "UNK". The BeerAdvocate dataset contains user reviews about beer from October 31, 2000 to January 11, 2012. The type of beer is chosen as category, and a user review is treated as an action. We use actions from October 31, 2000 to January 28, 2011 as the training set, those from January 28, 2011 to July 18, 2011 as the validation set, and the rest as the test set. We removed items with fewer than 5 actions, and removed users with fewer than 10 or more than 300 actions. Again, we merged categories with fewer than 100 items into the "UNK" category. The basic statistics of datasets are reported in Table 1.

Implementation Details. On both datasets, the in-category encoder of CoCoRec utilizes at most $T=20$ actions of the same category. The context encoder utilizes the most recent $L=20$ actions as input to both the top-k gating network and the recency encoder. The item input embedding layer shares the same parameters as the item output embedding layer. The category input embedding layer also shares the same parameters as the category output embedding layer. The self-attention networks in the in-category encoder, the top-k gating network and recency encoder stack $n_l = 2$ layers of a multi-head attention block with $n_h=1$ head and a feed-forward network block. In the objective function, the hyper-parameter λ is set to 0.5. The dropout rates are all set to 0.2. The batch size is set to 256. We utilize Adam as the optimizer.

On Taobao dataset, the dimension d_{in} of item embeddings is chosen from $\{128, 256, 512\}$. We set $d_{in}=256$ in our experiments, as we did not observe further improvement of performance with higher dimensions. The dimension d_h^c of the hidden representations in the self-attention network of the in-category encoder is chosen from $\{128, 256, 512\}$ and we set $d_h^c = 256$. The dimension d_{in}^z of category embeddings is chosen from $\{32, 64, 128\}$ and we set $d_{in}^z=64$ due to its promising performance. The dimension d_h^z of the hidden representations in the self-attention network of the top-k gating network is set to be the same as d_{in}^z. The top-k gating network selects $k = 5$ category-specific action subsequences for the next item prediction. Likewise, we set the dimension d_h^r in the self-attention network of the recency encoder to 256. The collaboration module records the in-category preferences of last $F=10240$ users and retrieves $f=256$ neighboring users. The number of negative items N_s in model training is set to 10000. We find that CoCoRec is sensitive to the learning rate, and the optimal

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[2] https://www.beeradvocate.com/

Table 1: Statistics of two evaluation datasets.

Dataset	#Users	#Items	#Categories	#Actions	#Actions per user	#Actions per item
Taobao	51,275	68,007	201	3,785,961	73.84±47.44	77.07±86.83
BeerAdvocate	7,313	17,373	102	563,638	55.67±69.44	32.44±98.76

learning rate 0.0001 is chosen from $\{0.0001, 0.0005, 0.001, 0.00001\}$. On BeerAdvocate dataset, the details of our model are as follows: $d_{in}=64, d_h^c=64, d_{in}^z=32, d_h^z=32$ and $d_h^r=64$. The hyper-parameters in collaboration module are $F=2048$ and $f=128$. The top-k gating network selects $k=5$ category-specific action subsequences for the next item prediction. The number of negative items N_s in model training is set to 1000. The learning rate is set to 0.00001. The influence of the hyper-parameters on the performance of CoCoRec is discussed later. The code has been released publicly [3].

4.2 Comparison against Baselines

We compare CoCoRec with an extensive set of baselines, and categorize them based on their modeling assumptions and the information they leveraged.

Heuristic solutions. We include two heuristic-based solutions, which have been shown to be strong baselines [29].

• *Global Popularity (GlobalPop).* It ranks items according to their popularities in the training set in a descending order.

• *Sequence Popularity (SeqPop).* It ranks items according to their popularities in the target user's sequence in a descending order. The popularity of an item is updated sequentially when more actions are observed.

Classical sequential recommendation solutions. We include solutions which only consider sequential order of actions for dependence modeling.

• *Recurrent Neural Network (GRU4Rec).* It adopts GRU for sequential recommendations [8].

• *Bidirectional Self-attentive Sequential Recommendation (BERT4Rec).* It adopts self-attention for sequential recommendations [27], which extends the SASRec model [13].

• *Multi-temporal-range Mixture Model (M3R).* It utilizes a mixture of RNN and self-attention to capture the dependence on both distant past actions and recent past actions for sequential recommendations [28].

Category-aware sequential recommendation solutions. We include solutions leveraging the item category for recommendations.

• *Category-Based Recommender (RNN+MTL).* It incorporates category information by treating it as an additional input to the neural sequence model [38]. Particularly, this solution utilizes multi-task learning to predict both the next item and its category.

• *Multi-Factor Generative Adversarial Network (MFGAN)*[4]. It utilizes adversarial training to ensure the generated action sequences still follow the distribution of the real-world action sequences [22]. It leverages the category of the next action and item popularity as conditions used by the discriminator in the adversarial learning.

Collaborative learning based sequential recommendation solutions. We include sequential recommendation solutions utilizing collaborative learning to combat data sparsity issue.

• *Collaborative Session-based Recommendation Machine (CSRM).* It utilizes the entire action sequences of users to define the users' similarities and retrieves neighboring users with similar preferences [34]. The neighboring users' preferences are combined with the target user's preferences to predict the next item.

• *Intent-guided Collaborative Machine for Session-based Recommendation (ICM-SR).* This work is an extension of CSRM, aiming at reducing the search space of neighbors [19]. It utilizes action sequences to make initial next item predictions and filters the irrelevant users based on these initial item predictions.

Evaluation Metrics. Recall@K and Mean Reciprocal Rank@K (MRR@K) are used as evaluation metrics. We rank all items for evaluation, instead of sampling a subset of items. This can avoid the bias introduced by the sampling to the evaluation, as Krichene and Rendle [16] demonstrated.

• *Recall@K*: It counts the proportion of times when ground-truth items are ranked among the top-K predictions.

• *MRR@K*: It reports the average of reciprocal ranks of the ground-truth items among the top-k predictions. If the rank is larger than K, the reciprocal rank is set to 0.

Results & analysis. The results of CoCoRec and baselines on two datasets are reported in Table 2, where CoCoRec outperformed all baselines. Regarding *heuristic solutions*, since GlobalPop treats each action independently and SeqPop only considers the dependence on actions associated with the same item, both of them performed worse than CoCoRec.

None of the *classical sequential recommendation solutions* leverage category information when modeling the action sequences. Without considering the fine-grained dependence among actions, vanilla applications of existing neural sequence models, like GRU4Rec and BERT4Rec, performed much worse than CoCoRec. M3R considers fine-grained dependence through a mixture of these neural sequence models. But its worse performance against CoCoRec suggests that leveraging category information is more useful to calibrate the dependence modeling than blindly combining a set of distinct neural sequence models.

The *category-aware sequential recommendation solutions* leverage category information to enhance the sequential recommendations. The worse performance achieved by RNN+MTL suggests that without careful design, considering category information does not necessarily improve sequential recommendation. MFGAN utilizes the category of the next action and the popularity of items to calibrate the dependence on past actions. But it does not specifically model item-to-item transition patterns within each category. Moreover, CoCoRec also makes use of category-specific action subsequences of the neighboring users to enhance the next item prediction. Therefore, we observe that CoCoRec achieved better performance than these category-aware solutions.

The *collaborative learning based sequential recommendation solutions* take advantage of neighboring users' actions to enhance the sequential recommendations. The better performance achieved by

[3]code available at https://github.com/RenqinCai/CoCoRec
[4]the knowledge graph in this method is omitted as it is not available on these two datasets.

Table 2: Performance of models on two datasets.

Models	Taobao				BeerAdvocate			
	Recall@5	MRR@5	Recall@20	MRR@20	Recall@5	MRR@5	Recall@20	MRR@20
GlobalPop	0.0024	0.0014	0.0076	0.0019	0.0078	0.0036	0.0341	0.0059
SeqPop	0.0944	0.0533	0.1754	0.0613	0.0004	0.0002	0.0016	0.0003
GRU4Rec	0.1283	0.0811	0.1888	0.0839	0.0173	0.0089	0.0492	0.0097
BERT4Rec	0.1291	0.0813	0.2122	0.0869	0.0222	0.0092	0.0533	0.0124
M3R	0.1294	0.0818	0.2163	0.0875	0.0235	0.0102	0.0615	0.0142
RNN+MTL	0.1283	0.0801	0.1979	0.0833	0.0202	0.0101	0.0573	0.0121
MFGAN	0.1307	0.0817	0.2176	0.0852	0.0233	0.0114	0.0599	0.0143
CSRM	0.1291	0.0818	0.1923	0.0842	0.0188	0.0093	0.0514	0.0112
ICM-SR	0.1299	0.082	0.2057	0.0849	0.0214	0.011	0.0587	0.0129
CoCoRec	**0.1557**	**0.0917**	**0.2609**	**0.1029**	**0.0278**	**0.0141**	**0.0737**	**0.0192**

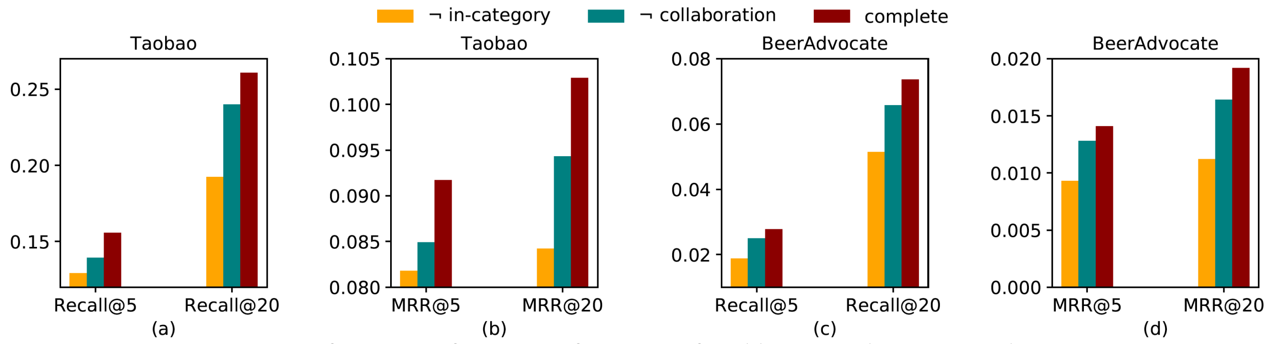

Figure 4: Performance of variants of CoCoRec for ablation analysis on two datasets.

CSRM over GRU4Rec shows explicit collaborative learning helps sequential recommendation. The comparison between CSRM and ICM-SR suggests that improving the quality of retrieved neighboring users enhances the utility of collaborative learning on sequential recommendations. Moreover, CoCoRec outperformed ICM-SR, which proves that context-aware neighborhood modeling can further increase the benefits of leveraging neighboring users' actions.

4.3 Detailed Analysis on Our Approach

4.3.1 Ablation Analysis. We conducted ablation analysis to demonstrate the importance of different components in CoCoRec. Specifically, we tested the following variants of CoCoRec:

¬ *in-category.* This variant excludes the in-category encoder from CoCoRec. It only utilizes the recency encoder and the collaboration module (ignoring item category) to predict the next item. The comparison between this variant and CoCoRec demonstrates the importance of using the in-category encoder to model the in-category user preferences.

¬ *collaboration.* This variant excludes the collaboration module from CoCoRec. It only uses the in-category encoder and the context encoder to predict the next item. The comparison between this variant and CoCoRec demonstrates the value of using the collaboration module to leverage neighboring users' in-category preferences.

Results are reported in Figure 4. The observation that these two variants perform worse than CoCoRec suggests that both the modeling f in-category preferences and the collaborative learning are useful for sequential recommendation.

4.3.2 Hyper-parameter Analysis. We changed the value of hyper-parameters in CoCoRec and investigated their influence on the recommendation quality.

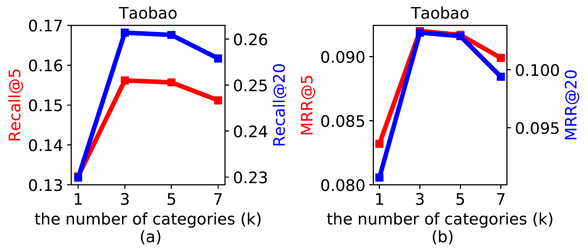

Figure 5: Performance of CoCoRec with different number of selected categories (k) on Taobao.

• *Number of categories in top-k gating network.* Because of the influence from the next category prediction on the next item prediction, we study the effect of the number of categories k selected by the gating network on the performance of CoCoRec. We report results in Figure 5. We can observe CoCoRec achieves the best performance with $k = 3$; and when $k = 1$, the performance drops. This is caused by imperfect category prediction, i.e., the ground-truth categories are not ranked at the first position. In particular, the performance of the category prediction is Recall@5:0.7328 and MRR@5:0.5510. The wrong predictions of the next category further impact the next item predictions. On the other hand, when k keeps increasing, the performance also drops. The reason is that larger k means more in-category user preferences are included for the next item prediction. This inevitably introduces irrelevant in-category preferences, which eventually hurts the next item prediction.

• *Number of actions in in-category encoder and context encoder.* The lengths of category-specific action subsequences affect the modeling of in-category user preferences. We study the effect of the number of actions T per category on the performance of CoCoRec. In addition, since the number of recent actions affects both the modeling of category context and episodic context, we also study the effect of the number of recent actions L on the performance of CoCoRec. The results on Taobao dataset are reported in Table 3. We can observe when too few actions are considered in the in-category encoder, i.e., $T = 2$, CoCoRec perform poorly. This is because limited information about the in-category user preferences can be captured. Similarly, when too few recent actions are considered in the context encoder, i.e., $L = 2$, the performance drops. This suggests recent actions contain important signals for the next item prediction.

Table 3: Performance of CoCoRec with different number of actions in in-category encoder (T) and context encoder (L) on Taobao dataset.

Settings		Recall@5	MRR@5	Recall@20	MRR@20
T=2	L=20	0.1322	0.0821	0.2371	0.0931
T=20	L=20	0.1557	0.0917	0.2609	0.1029
T=50	L=20	0.1559	0.0918	0.2604	0.1026
T=20	L=2	0.1301	0.0811	0.2253	0.0876
T=20	L=50	0.1559	0.0919	0.2612	0.1030

• *Number of retrieved neighboring users in collaboration module.* Because the number of retrieved neighbors affects the collaborative learning, we study the effect of the number of retrieved neighboring users f on the performance of CoCoRec. The results are reported in Table 4. We can see retrieving either too few or too many neighboring users hurts the next item prediction. When too few neighbors are retrieved, the collaboration effect is limited. Thus, the CoCoRec cannot benefit from similar users. On the other hand, when too many neighbors are retrieved, including the action subsequences from less relevant users hurts the next item prediction.

Table 4: Performance of CoCoRec with different number of retrieved neighbors on Taobao dataset.

Settings	Recall@5	MRR@5	Recall@20	MRR@20
f=128	0.1502	0.0903	0.2371	0.0931
f=256	0.1557	0.0917	0.2609	0.1029
f=1024	0.1534	0.0907	0.2583	0.1017

4.3.3 *Case Study.*

To examine our model's behaviors, we qualitatively study in-category action subsequences, recent action subsequences, neighors' in-category action subsequences, and the attention weights of actions in these subsequences. Due to the space limit, we select a user and one of her actions in the testing set as the target action. We retrieve the neighbors with similar in-category preferences to her, as Figure 6 shows. We can observe the in-category subsequence of the top ranked neighbor has actions in common with the in-category subsequence of the target user.

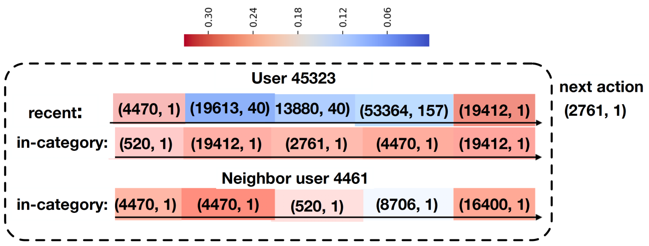

Figure 6: An example of the next item prediction by CoCoRec. The target user's (user 45323) action sequence is separated into category-specific subsequences. On the top, We visualize the action subsequence containing the recent actions, and the action subsequence specific to the category of the next item. On the bottom, the most similar neighbor's (user 4461) in-category subsequence is presented. The actions in each subsequence are indexed chronologically, denoted as a tuple (item id, category id). The color indicates the attention weight in self-attention networks.

The actions associated with items (e.g., item 19412) which have appeared multiple times likely have large attention weight. In addition, the item-to-item transitions (from item 19412 to item 2761) in the category-specific subsequence suggest the next item.

5 CONCLUSION

In this paper, suggested by our statistical analyses on the dependence structure introduced by the category-specific action context, we propose CoCoRec to leverage category information to capture the context-aware action dependence for sequential recommendation. Specifically, a sequence of actions is decomposed into multiple subsequences with respect to item categories. Within each category-specific subsequence, CoCoRec employs a self-attention network to capture item-to-item transition patterns. A top-k gating network is employed to predict the category of the next item, so as to activate the in-category preferences for the next item prediction. Besides, CoCoRec models the most recent actions as episodic context, due to their close proximity to the next action. To handle sparsity in individual user's action sequences, CoCoRec employs context-aware collaborative learning across users with similar in-category preferences. Extensive experiments and ablation analyses on two large datasets show that category-aware dependency modeling and context-aware collaborative learning in CoCoRec help improve its sequential recommendation quality.

Currently we only considered the sequential order of actions in users' interaction history with the system. Since the actual time of those actions also conveys important contextual information, it is important to consider how to extend CoCoRec to model such temporal information in the furture work.

ACKNOWLEDGMENTS

We would like to thank Zhan Xu, Haiyun Jin and Ruocheng Guo for insightful feedback and discussions. We also thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported in part by the US National Science Foundation under grants IIS-2007492, IIS-1553568 and IIS-1718216.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender systems handbook*. Springer, 217–253.

[2] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural News Recommendation with Long-and Short-term User Representations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*. 336–345.

[3] Ron Begleiter, Ran El-Yaniv, and Golan Yona. 2004. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research* 22 (2004), 385–421.

[4] Renqin Cai, Xueying Bai, Zhenrui Wang, Yuling Shi, Parikshit Sondhi, and Hongning Wang. 2018. Modeling Sequential Online Interactive Behaviors with Temporal Point Process. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 873–882.

[5] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. ACM, 108–116.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 223–231.

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Hao Hou and Chongyang Shi. 2019. Explainable Sequential Recommendation using Knowledge Graphs. In *Proceedings of the 5th International Conference on Frontiers of Educational Technologies*. 53–57.

[11] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 573–581.

[12] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 306–310.

[13] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[14] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. 79–86.

[15] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 447–456.

[16] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[17] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1419–1428.

[18] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 322–330.

[19] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An Intent-guided Collaborative Machine for Session-based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1833–1836.

[20] Jiarui Qin, Kan Ren, Yuchen Fang, Weinan Zhang, and Yong Yu. 2020. Sequential recommendation with dual side neighbor-based collaborative relation modeling. In *Proceedings of the 13th international conference on web search and data mining*. 465–473.

[21] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[22] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 89–98.

[23] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[25] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1 (2001), 285–295.

[26] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 555–563.

[27] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv preprint arXiv:1904.06690* (2019).

[28] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H Chi. 2019. Towards neural mixture recommender for long range dependent user sequences. In *The World Wide Web Conference*. ACM, 1782–1793.

[29] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 565–573.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[31] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Make it a chorus: knowledge-and time-aware item modeling for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 109–118.

[32] Jianling Wang and James Caverlee. 2019. Recurrent Recommendation with Local Coherence. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 564–572.

[33] Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee, and Liangjie Hong. 2020. Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 645–653.

[34] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 345–354.

[35] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2467–2475.

[36] Jibang Wu, Renqin Cai, and Hongning Wang. 2020. Déjà vu: A Contextualized Temporal Attention Mechanism for Sequential Recommendation. In *Proceedings of The Web Conference 2020*. 2199–2209.

[37] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems. In *The World Wide Web Conference*. ACM, 2236–2246.

[38] Qian Zhao, Jilin Chen, Minmin Chen, Sagar Jain, Alex Beutel, Francois Belletti, and Ed Chi. 2018. Categorical-Attributes-Based Multi-Level Classification for Recommender Systems. (2018).

[39] Nengjun Zhu, Jian Cao, Yanchi Liu, Yang Yang, Haochao Ying, and Hui Xiong. 2020. Sequential Modeling of Hierarchical User Intention and Preference for Next-item Recommendation. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*. ACM.