# Modeling Multiple Coexisting Category-Level Intentions for Next Item Recommendation

YANAN XU, YANMIN ZHU, and JIADI YU, Shanghai Jiao Tong University

Purchase intentions have a great impact on future purchases and thus can be exploited for making recommendations. However, purchase intentions are typically complex and may change from time to time. Through empirical study with two e-commerce datasets, we observe that behaviors of multiple types can indicate user intentions and a user may have multiple coexisting category-level intentions that evolve over time. In this article, we propose a novel Intention-Aware Recommender System (IARS) which consists of four components for mining such complex intentions from user behaviors of multiple types. In the first component, we utilize several Recurrent Neural Networks (RNNs) and an attention layer to model diverse user intentions simultaneously and design two kinds of Multi-behavior GRU (MGRU) cells to deal with heterogeneous behaviors. To reveal user intentions, we carefully design three tasks that share representations from MGRUs. The next-item recommendation is the main task and leverages attention to select user intentions according to candidate items. The remaining two (item prediction and sequence comparison) are auxiliary tasks and can reveal user intentions. Extensive experiments on the two real-world datasets demonstrate the effectiveness of our models compared with several state-of-the-art recommendation methods in terms of hit ratio and NDCG.

CCS Concepts: • **Information systems** → **Recommender systems**; Learning to rank; • **Applied computing** → *Online shopping*;

Additional Key Words and Phrases: Recurrent neural networks, recommender system

**ACM Reference format:**
Yanan Xu, Yanmin Zhu, and Jiadi Yu. 2021. Modeling Multiple Coexisting Category-Level Intentions for Next Item Recommendation. *ACM Trans. Inf. Syst.* 39, 3, Article 23 (May 2021), 24 pages.
https://doi.org/10.1145/3441642

## 1 INTRODUCTION

Recommender systems have become an essential component of online retail websites and aim to recommend items that a user will most likely to purchase in the future. With the fast development of e-commerce, these websites recorded a massive amount of heterogeneous interaction behaviors

This research is supported in part by the 2030 National Key AI Program of China 2018AAA0100503 (2018AAA0100500), National Science Foundation of China (No. 62072304, No. 61772341, No. 61472254, No. 61770238), Shanghai Municipal Science and Technology Commission (No. 18511103002, No. 19510760500, and No. 19511101500), the Program for Changjiang Young Scholars in University of China, the Program for China Top Young Talents, the Program for Shanghai Top Young Talents, SJTU Global Strategic Partnership Fund (2019 SJTU-HKUST), the Oceanic Interdisciplinary Program of Shanghai Jiao Tong University (No. SL2020MS032) and Scientific Research Fund of Second Institute of Oceanography (No. SL2020MS032).
Authors' address: Y. Xu, Y. Zhu (corresponding author), and J. Yu, Shanghai Jiao Tong University, 800 Dongchuan RD. Minhang District, Shanghai, China, 200240; emails: {xuyanan2015, yzhu, jiadiyu}@sjtu.edu.cn.

1046-8188/2021/05-ART23 $15.00
https://doi.org/10.1145/3441642

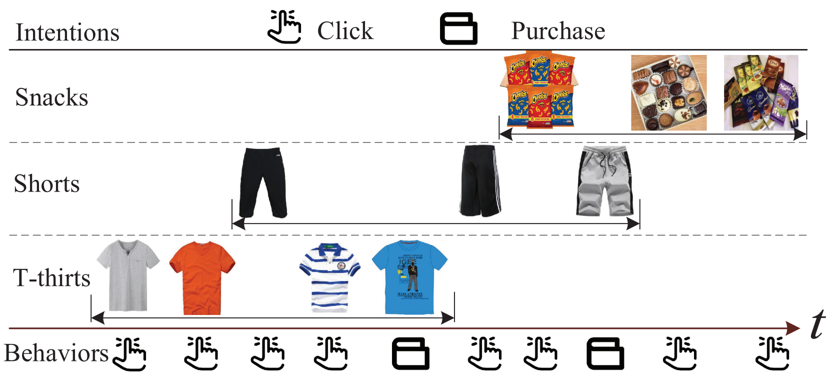


Fig. 1. An example of multiple coexisting category-level intentions indicated by interaction behaviors of a user.

(e.g., view, cart, and purchase) between users and items. These interaction records provide opportunities for understanding user purchase intentions.

Along this line, recommendation with data of behaviors of multiple types (*multi-behavior data* for short) has attracted wide concerns. Existing studies leveraging multi-behavior data can be divided into two main categories. The first category is extending matrix factorization models. These studies aim to learn the relevance between users and items based on multiple interaction matrices [7, 9, 18, 34]. However, these methods take user-item relationships into consideration from static views and neglect the evolution of users' intentions and the context of purchase behavior. The second kind of work recommends next items by mining sequential patterns or modeling dependency between items [17, 35]. In these studies, researchers show more interest in mining the user intentions in short sessions for recommendation. But they model behaviors of different types independently and convert a sequence to a single latent vector which may fail to handle complex intentions.

In this article, we aim to exploit the multi-behavior data to mine user purchase intentions for the next-item recommendation. *Behaviors* are the ways (e.g., click, add-to-cart, purchase) that users interact with items on websites. Users usually have multiple kinds of behaviors (e.g., click many related items) before they finally buy a product. We find that behaviors of multiple types can indicate user intentions and users may have multiple coexisting purchase intentions on items of several categories through an empirical study (see details in Section 3). Intentions denote the purpose or goal of one purchase interaction. For example, a user wants to buy both a T-shirt and a pair of shorts (i.e., two kinds of intentions) when summer comes (see Figure 1). He may have click behaviors on products of the two categories alternately before he buys them. Therefore, users can have more than one intention simultaneously. If we can learn such latent complex user intentions from multi-behavior data with item category information and dynamically determine which intention is the most important for the purchase of next item, we will improve the recommendation performance greatly. For better modeling such multiple coexisting intentions based on interaction behaviors, our proposed model should satisfy the following several characteristics:

- *Exploiting Different Behaviors Discriminately.* Behaviors of various types have different importance for indicating user intentions. Casual viewing behaviors may be generated with curiosity while purchasing behaviors usually confirm users' preference on items. Therefore, they should be exploited discriminately.

- *Reserving the Dependency among Behaviors of Diverse Types.* On the one hand, the interaction data should be seen as sequences for retaining the dependency among items and evolving trends of user intentions. On the other hand, behaviors of different types should be seen as the context of adjacent behaviors rather than be utilized independently.
- *Modeling Multiples User Purchase Intentions Simultaneously.* Previous research presents user intentions with a single latent vector which lacks the ability for modeling multiple intentions. Our approach should learn several intentions on item category levels simultaneously. Previous research usually mines user hidden intentions from interaction sequences of item IDs. But categories of items in interaction records can explicitly reflect user intentions which need to be further studied.

To achieve the above goals, we propose a novel neural network model named *IARS* to model user purchase intentions with the sequences of multiple kinds of behaviors. The model consists of four components, i.e., one RNN-based module and three task modules. The RNN-based module is used as one encoder to learn latent representations encoding user intentions from sequences of behaviors. It discriminately processes different behaviors using two proposed *Multi-behavior Gated Recurrent Unit (MGRU)* cells. In addition, for modeling users' multiple intentions, we utilize several RNNs to process behaviors simultaneously and adopt attention in recommendation to enable each RNN to learn different intentions. The other three modules are designed for solving diverse tasks based on the learned representations. The first task is to predict next item and category. The second task is to judge whether two sequences are of the same user and whether they are adjacent sequences. These two tasks are aimed to reveal user intentions from record levels and sequence levels, respectively. As far as we know, we are the first to propose sequence comparison tasks to improve recommendation performance. The third task is to estimate user preference over items which is the main task of our model. In this task, we utilize an attention layer to select user intentions according to candidate items.

To summarize, the main contributions of this article are as follows:

- We conduct an empirical study on two real-world e-commerce datasets and observe that users have multiple coexisting category-level intentions in interaction sequences of diverse behaviors.
- We propose a novel neural model named IARS to mine user complex intentions for recommendation. It includes several recurrent neural networks for learning latent multiple user intentions from behaviors and three diverse tasks. The various auxiliary behaviors are utilized to reveal user intentions before purchase behavior. Item category prediction task and sequence comparison task are proposed to further reveal user intentions.
- We conduct experiments on the two real-world datasets. The results show that our model significantly outperforms other state-of-the-art methods from various aspects.

The remainder of this article is organized as follows: In Section 2, we first introduce related work. In Section 3, we analyze two real-world datasets and show several key observations. Next, we introduce our proposed models in Section 4. Then, we evaluate our models based on the two e-commerce datasets in Section 5. Finally, we conclude this article.

## 2 RELATED WORK

### 2.1 Multi-Behavior Recommendation

In recent years, it has been very popular in the recommendation area to study users' long-term and short-term interests from interaction sequences for improving recommendation performance [20, 32]. Ma et al. [20] proposed a hierarchical gating network to capture both long-term and short-term user interests with feature gating and instance gating modules. They also captured relations between items with an item-item product module. Zhang et al. [32] utilized self-attention to learn item-item relations as short-term user interests and assigned each user a latent vector to model long-term preference. The above two studies both learn users' long-term and short-term interests, but their models can only deal with datasets having a single kind of behavior. Sun et al. [27] modeled user interaction sequences with the deep bidirectional self-attention network BERT which neglects the order of interaction records and is trained by randomly masked some items as inferred targets. Wang et al. [31] proposed a mixture-channel purpose routing network to learn users' complex purposes from interaction sequences and assigned interaction records into the different purpose channels using a routing function. This work can learn users' multiple intentions. But it can not deal with multi-behavior data and does not take advantage of category information which is proved very useful in our experiments. In addition, its routing function is not continuous and may make it hard to learn parameters with gradient descent methods.

Multi-behavior recommendation utilizes interaction data of multiple kinds of behaviors and improves recommendation performance of one target behavior with other types [9, 17, 18, 26]. Some work extended matrix factorization to deal with the multi-behavior scenarios [16, 26, 34]. Singh and Gordon [26] designed a *Collective Matrix Factorization (CMF)* model by simultaneously factorizing several user-item matrices and sharing latent factors of items. The CMF model is further extended to deal with datasets of various behaviors [16, 34]. Krohn-Grimberghe et al. [16] proposed to factorize both a user-item matrix from interactions and a user-user matrix from social networks, and share user embeddings. Zhao et al. [34] combined user-topic matrices of different behavior groups and built user topic profiles by factorizing these matrices.

Some other work deals with multi-behavior data from the perspective of learning [7, 9, 18, 23]. Loni et al. [18] proposed a multi-channel Bayesian Personalized Ranking (BPR) which samples negative items from different behaviors with different sampling rules. Qiu et al. [23] proposed BPRH model for heterogeneous implicit feedbacks considering the correlation between auxiliary action and target action with co-occurrence of them. Wan and McAuley [29] proposed a structure called *monotonic behavior chains* to describe that strong signals can imply the presence of a weaker signal, and designed a model named *chainRec* to learn such dependencies among different behaviors. Loni et al. [19] treated multiple kinds of behaviors as channels and provided different sampling methods to sample negative instances for training an FM model. Ding et al. [7] proposed a margin-based pairwise learning model and assigned different margins for different behaviors. Chen [3] built a Behavior2Vec model to generate distributed representation for users' multiple behaviors on products. Gao et al. [9] designed a cascaded neural network to exploit user behaviors of multiple types and treated each kind of behavior as one task for recommendation.

The above research considers user behaviors of multiple types, but omits the sequential dependency between behaviors. For exploiting multiple behaviors and behavior dependency, Li et al. [17] proposed to model different behaviors with different RNNs. However, they omit the dependency between different kinds of behaviors. Chen et al. [4] proposed a method called AIR which consists of two steps. They first predicted item category for obtaining user intention representation and then recommended next interacted item based on user intentions with an FM model. But AIR is not an end-to-end model which can limit its performance. Zhou et al. [35] projected behaviors of different types into multiple latent semantic spaces for modeling users' complex interests and the influence among behaviors is modeled via self-attention. But this model processes interaction records independently and can not learn the sequential patterns in sequences. Li et al. [17] first learned item embeddings with multi-behavior sequences using an item2vec model. Then, they divided behaviors into auxiliary behaviors and a target behavior, and processed them with two LSTMs to learn users' current motivation and historical preference, repectively. However, this method does not consider that users may have several interests or motivations over a period of time. Zhu et al. [36] proposed to build an interactive recommender system to mine user intentions by asking users questions and exploits their various feedbacks. Their model relies on dynamically interacting with users and the problem formulation is quite different from ours.

Tanjim et al. [28] used a Temporal Convolutional Network layer to capture users' latent intent from sequences of item categories and guide an attentive model with learned intentions to predict the next item. The proposed model can deal with various behaviors and item category information. But the model can not process different behaviors discriminately.

### 2.2 Multi-task Learning

The high performance of machine learning methods depends on good data representations to a great extent [1]. Representation learning of a single task can only capture the information needed by the task and discarded other information. In contrast, multi-task learning allows statistical strength sharing and knowledge transfer, thus the representations can capture more underlying factors and have higher generalization abilities [2]. In some studies of multi-task learning, researchers treat tasks equally and try to improve the performance of all tasks. Chen et al. [5] proposed an adversarial multi-criteria learning method for solving the Chinese word segmentation task and designed an adversarial strategy to force shared network layers to learn criteria-invariant features. In other research, tasks are divided into one main task and other auxiliary tasks. The auxiliary tasks are utilized for improving the performance of the main tasks. Seltzer and Droppo [25] performed a primary classification task and other auxiliary tasks by sharing representations in a deep neural network, and proved that multi-task learning can reduce greatly the classification errors. Zhang et al. [33] proposed to solve the problem of facial landmark detection together with head pose estimation and facial attribute inference to improve detection performance.

In recommendation areas, many studies utilize multi-task learning framework [15, 22]. Wang et al. [30] utilized a knowledge graph embedding task to assist recommendation task and associated two tasks with cross&compress units to share latent features between items and entities. Huang et al. [13] studied the problem of entity recommendation and web page retrieval by sharing context representations based on search logs from search engines. Elkahky et al. [8] studied the cross-domain recommendation problem and their model shared user representation across different domains. Their work can also be seen as multi-task research and treated recommendations in each domain as one task. Multi-behavior data in the recommendation domain can also be utilized to construct multiple tasks and each behavior is treated as one task [9, 34]. Zhu et al. [37] used brand information to improve performance.

## 3 PRELIMINARIES

In this section, we first introduce two datasets and empirical studies on them. Then, we formally define the next-item recommendation task with multi-behavior data.

### 3.1 Data Analysis

We utilize two e-commerce datasets, i.e., **Taobao** and **Retailrocket**, which contain multiple kinds of user behaviors. The two datasets are collected from Taobao app and a retail website, respectively. The statistics of the two datasets are shown in Table 1. The descriptions of the datasets are shown as follows:

- **Taobao**[1] dataset is collected on the Taobao app which belongs to the largest electric business company (i.e., Alibaba) in China within the time period from 2014/11/18 to 2014/12/18.

[1]https://tianchi.aliyun.com/dataset/dataDetail?dataId=46.

Table 1. Statistics of Datasets

| Dataset | Description | Value | Dataset | Description | Value |
|---|---|---|---|---|---|
| Taobao | # users | 10,000 | Retailrocket | # users | 1,407,580 |
| | # items | 2,876,947 | | # items | 417,053 |
| | # categories | 8,916 | | # categories | 1,086 |
| | Click | 11,550,581 | | View | 2,664,312 |
| | Collect | 242,556 | | Cart | 69,332 |
| | Cart | 343,564 | | Purchase | 22,457 |
| | Purchase | 120,205 | | | |

This dataset contains four kinds user behaviors including *click*, *collect*, *cart*, and *purchase*. The total number of all interactions is 12, 256, 906.

- **Retailrocket**[2] is from a real online retail website within the time period from 2015/05/03 to 2015/09/18. It records three kinds of behaviors including *view*, *cart*, and *purchase*. The number of interactions is 2, 756, 101.

According to the statistic information in Table 1, we observe that the datasets both have item category information and have a large number of click/view records though they are not the most concerning interactions for e-commerce companies. The findings inspire us whether we can utilize the two kinds of information. We conduct two empirical studies on the datasets.

We first show the distribution of interaction records belonging to a user or an item. The results are shown in Figure 2. It should be noticed that axes are in log scales. We can see that (1) *for all kinds of behaviors, the number of records belonging to one user/item follows power-law distributions.* (2) *The number of records for click/view is much larger than that of other behaviors.* Retailrocket is sparser than Taobao dataset.

Next, we study users' intentions over item categories in behaviors. In daily life, people usually search for products with categories according to their demands. They will compare several items of the same categories and buy a suitable one. It inspires us that the recent behaviors, especially click/view, can indicate users' intentions for buying products. In this experiment, we sort interaction records of each user by timestamps and split them at the positions of purchase behavior records (see Figure 3 as one example). As a result, each user has several interaction sequences of items (e.g., $S_1$, $S_2$, and $S_3$). For each sequence, we can further split it into two subsequences of the same length (e.g., $S_4$ and $S_5$).

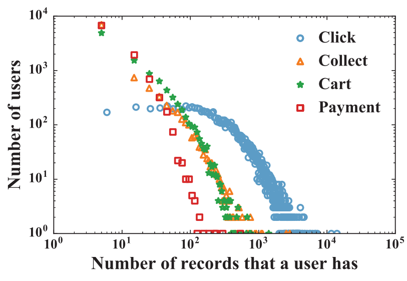
Then, we can calculate the distribution of items over categories in every sequence with the following equation:

$$p(c = i|S) = \frac{|\{r|r.c = i, r\ in\ S\}|}{|S|}, \tag{1}$$

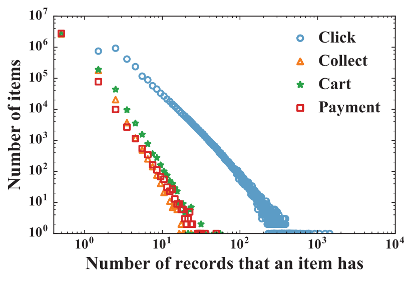
where $c$ denotes category, $r$ is one record in sequence $S$ and $|S|$ is the number of records in it. It should be noted that if one item was interacted with for more than one time in a sequence, we treat the interactions as different records because their timestamps are different. Based on the distributions, we can compute the Kullback–Leibler divergence (KL divergence) between arbitrary two sequences to show their similarities with Equation (2).

$$D_{KL}(S_j, S_k) = \sum_i p(c = i|S_j) \log \frac{p(c = i|S_j)}{p(c = i|S_k)}. \tag{2}$$

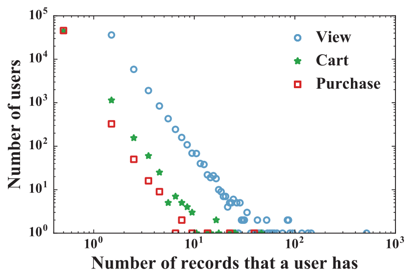
[2]https://www.kaggle.com/retailrocket/ecommerce-dataset.



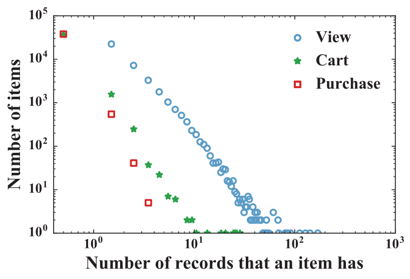
(a) The number of records a user has for Taobao.



(b) The number of records an item has for Taobao.



(c) The number of records a user has for Retail.



(d) The number of records an item has for Retail.

Fig. 2. Distributions of number of records that one user/item has.

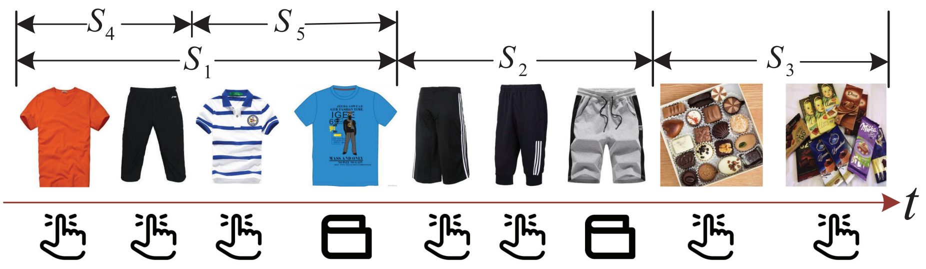
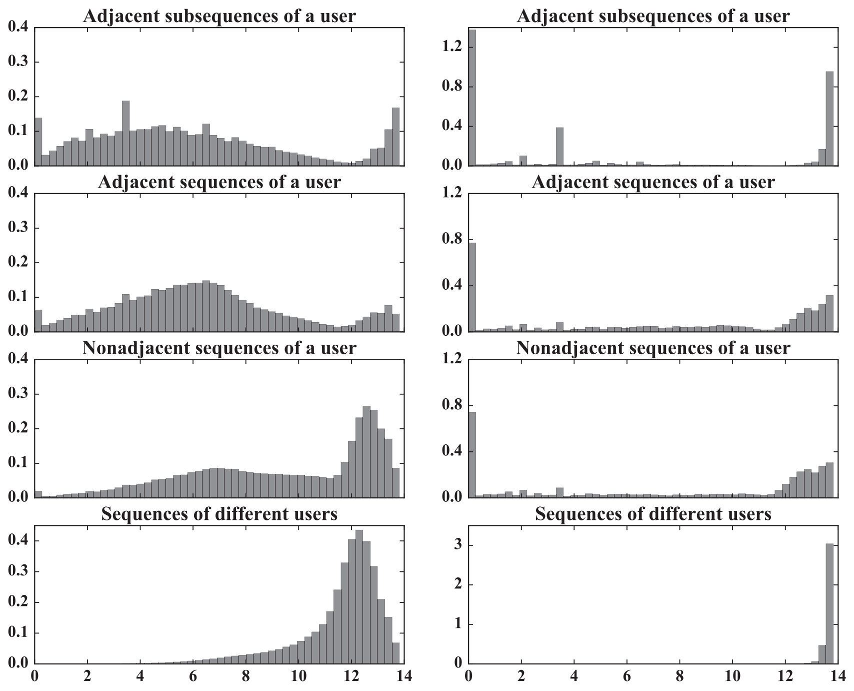


Fig. 3. Interaction records of a user. The figure shows two kinds of behaviors including click and purchase. The interaction records are divided into several sequences, i.e., $S_1$ to $S_5$, according to purchase records.

Considering KL divergence is asymmetrical, we further compute average value of $D_{KL}(S_j, S_k)$ and $D_{KL}(S_k, S_j)$ which is denoted by $D_K(S_j, S_k)$. When the KL divergence is smaller, the two sequences are more similar on item distribution over item categories. We consider four kinds of sequence pairs. For a user, we consider adjacent subsequences between two adjacent purchases ($S_4$ and $S_5$), adjacent sequences ($S_1$ and $S_2$), and nonadjacent sequences of a user ($S_1$ and $S_3$). The last kind of pair is the sequences of different users. It should be noted that adjacent subsequences must be two subsequences split from one sequence. The histograms of their KL divergences are shown in Figure 4. For each plot, the abscissa is the KL divergence and vertical is the number of sequence pairs. As the total number of sequence pairs is different in the four plots, the histograms are all normalized.

From the figures, we have several observations.

(1) *Users have purchase intentions.* The KL divergence of the first row is usually lower than that of the other three rows. It means that the interacted items between two purchased behaviors are



(a) Taobao dataset. (b) Retail dataset.

Fig. 4. Histograms of KL divergences between two interaction sequences in terms of item categories on two E-commerce datasets. For each user, his interaction records can form sequences of items and the interaction sequences are split according to purchase behaviors. From top to down, the first plot shows KL divergences between adjacent subsequences of the same users. The second plot shows KL divergences adjacent sequences separated by purchase behaviors of the same user. The third plot is KL divergence of two nonadjacent sequences of the same user. The fourth plot is divergences between sequences from different users. For each plot, the horizontal is KL divergence and the vertical is the number of sequence pairs.

usually very similar in terms of item categories. Thus, the plots in the first row have the smallest KL divergence (e.g., $D_K(S_4, S_5)$) than other plots. When the user has bought one item, his intention on the category of that item ends. Thus, the KL divergences become larger in the second row ($D_K(S_1, S_2) > D_K(S_4, S_5)$). As time goes on, users' intention changes and his intention is also very different from that of other users. As a result, the plots in the third row and fourth row have larger KL divergence.

(2) *Users may have multiple coexisting category-level intentions.* If a user has only one intent, his/her item sequences between two purchases should belong to one category. But the plots in the first row show that the KL divergences are usually larger than 0, e.g., $D_K(S_4, S_5) > 0$.

(3) *Users' intentions evolve over time.* Comparing the first three rows, we can see that the KL divergences grow with the time interval between two sequences. KL divergences of nonadjacent sequences are obviously larger than that of adjacent sequences and subsequences, e.g., $D_K(S_1, S_3) > D_K(S_1, S_2) > D_K(S_4, S_5)$. We also compute the average KL divergences for each kind of sequence pairs as shown in Table 2. We can see it clearly that from top to down, the mean values of KL divergences grow.

Table 2. The Average KL Divergences for Different Kinds of Sequence Pairs

| Dataset | Taobao | Retailrocket |
|---|---|---|
| Adjacent subsequences | 5.90 | 5.73 |
| Adjacent sequences | 6.33 | 7.40 |
| Nonadjacent sequences | 9.42 | 7.81 |
| Sequences of different users | 11.56 | 13.69 |

## 3.2 Problem Definition

Next-item recommendation is the task of predicting which item a user will interact with based on his/her historical interaction records. In what follows, we will present some notifications and formulate the recommendation problem.

Interactions with items of one user naturally form a sequence over time. Therefore, interaction history $H$ from an e-commerce platform can be seen as a set of interaction sequences, i.e., $H = \{S_1, S_2, \ldots, S_u, \ldots, S_N\}$, where $S_u$ denotes the interaction sequence of user $u$ and $N$ is the number of users. For a user $u$, his/her interaction sequence is $S_u = \{(x_1, c_1, b_1), (x_2, c_2, b_2), \ldots, (x_T, c_T, b_T)\}$, where $(x_t, c_t, b_t)$ is one interaction record, $x_t$ denotes an item, $c_t$ is the category of $x_t$, and $b_t$ denotes the behavior type (e.g., view, cart, purchase). We use $B = \{b_1, b_2, \ldots, b_R\}$ to denote the set of all behaviors and $R$ is the number of behavior types. As we aim to improve the recommendation performance of purchase, we treat purchase $b_R$ as the target behavior and the rest as auxiliary behaviors. Then, we can define the next-item recommendation problem.

Given a user $u$ and his/her interaction sequence $S_u = \{(x_1, c_1, b_1), (x_2, c_2, b_2), \ldots, (x_T, c_T, b_T)\}$ of all behavior types, we aim to recommend a new item $x$ that the user $u$ will most likely purchase at $T + 1$.

## 4 IARS: INTENTION-AWARE RECOMMENDER SYSTEM

In this section, we introduce our model, i.e., IARS, for the next-item recommendation based on multi-behavior data. We first provide an overview of our model and describe its main components. Finally, we show the training steps.

## 4.1 Overview of IARS

Before introducing the implementation details, we give an overview of our proposed model. As we have said in the previous section, we want to utilize rich data of various behaviors to mine user intentions. Our model should satisfy three requirements. First, it should retain the dependency between behaviors of multiple types in sequences and not process them independently. Second, it should model different behaviors discriminately as various behaviors have different importance. Third, it should capture multiple coexisting intentions at the same time. To meet these requirements, we propose a model named *Intention-Aware Recommender System (IARS),* whose structure is shown in Figure 5. It consists of four components, i.e., one encoder for mining user intentions from behaviors with multiple RNNs and the other three components as decoders for different tasks based on user intentions.

The encoder can encode a sequence of behaviors into several sequences of latent vectors. The encoder can satisfy all the requirements we mentioned. First, the interaction records of different kinds of behaviors are treated as one sequence for each user. The sequences of behaviors retain dependency among them. Second, we use behavior types as the input and propose two kinds of Multi-behavior GRU (MGRU) cells including Hard-MGRU and Soft-MGRU for RNNs to deal with behavior types. Third, we utilize several RNNs to capture multiple intentions of one user.

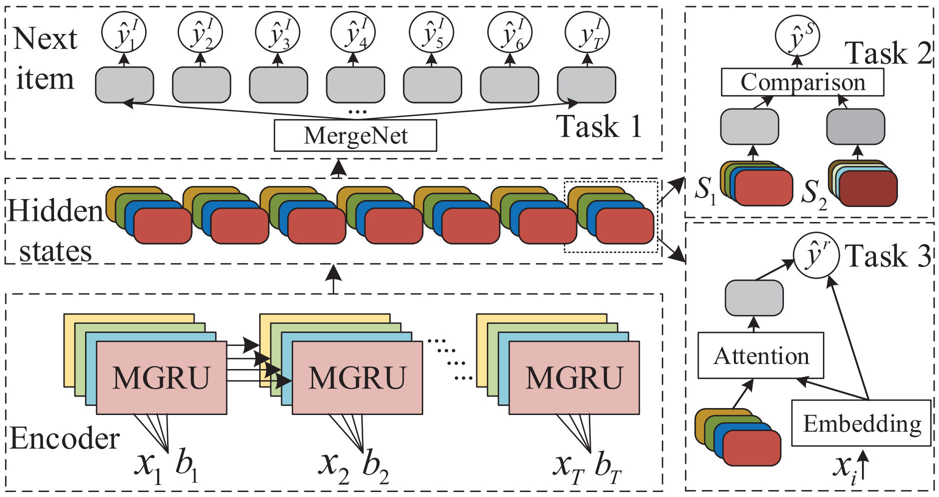

Next item
$\hat{y}_1^I$ $\hat{y}_2^I$ $\hat{y}_3^I$ $\hat{y}_4^I$ $\hat{y}_5^I$ $\hat{y}_6^I$ $\hat{y}_T^I$
MergeNet
Task 1
$\hat{y}^S$
Task 2
Comparison
$S_1$ $S_2$
Hidden states
$\hat{y}^r$
Task 3
Attention
Embedding
$x_i$
MGRU
Encoder
$x_1$ $b_1$ $x_2$ $b_2$ $x_T$ $b_T$

Fig. 5. The overview of our model. Multiple MGRU networks are denoted by different colors.

In addition, the categories of items are also used as the input of the encoder for learning users' intentions on categories.

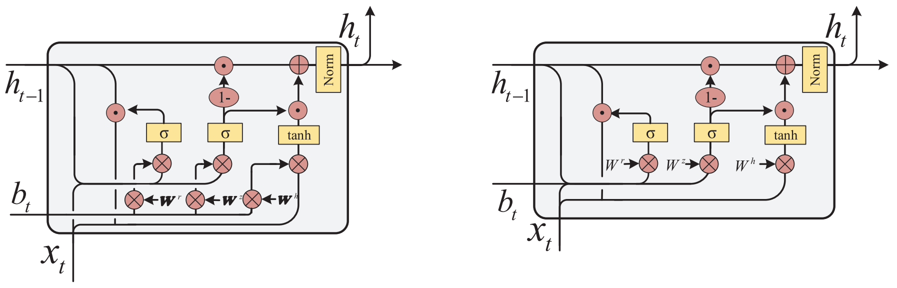
The sequences of latent vectors generated by RNNs contain information about user intentions and will be used for finishing three kinds of tasks. The first two tasks are used as auxiliary tasks to further reveal users' intentions and the third task is our main task, i.e., recommendation task. The first task is predicting the next item and item category. The second task is given two sequences, judge whether they are from the same user and whether they are adjacent sequences of the same user. In the third task, we use the last hidden states of MGRU as the representation of users and calculate his/her preference score on each item. What is more, in each task, we offer different kinds of methods to merge multiple hidden states from RNNs. For the recommendation task, we leverage an attention layer to merge multiple hidden states and let each RNN learn different intentions.

## 4.2 Hard-MGRU and Soft-MGRU

In this part, we introduce the encoder component. This component is proposed to detect users' intentions from behaviors of multiple types and retain the item dependency information in sequences. Inspired by [11], the encoder consists of several RNNs and each RNN is used to learn one kind of user intention using different parameters. For each RNN, its input includes sequences of behavior types ($b_t$), item IDs ($x_t$), and item category IDs ($c_t$).

To reserve the dependencies between records of various behavior types, we organize records of all behavior types in one sequence for each user, and process them using a unified MGRU network. The information of extracted user intentions is passed between adjacent records with hidden states of MGRU. To deal with the behavior types, we propose two kinds of MGRU cells, i.e., Hard-MGRU and Soft-MGRU. Hard-MGRU processes various behaviors with different parameter sets. It can extract different information from them and adapts to different behavior patterns (e.g., click records generate faster than purchase records). Soft-MGRU processes behavior types with gates to determine the importance of each kind of behavior. Considering behavior types contain little information about user intentions, we do not use them in the computation of candidate hidden states in Soft-MGRU.

As all RNNs in the encoder have identical structures but different parameters, we omit the notations for distinguishing them. In what follows, we will introduce Hard-MGRU and Soft-MGRU.



(a) Hard-MGRU (b) Soft-MGRU

Fig. 6. Structure of two kinds of MGRU cells. Embedding layers and item categories are omitted for simplicity. $\otimes$, $\odot$, and $\oplus$ denote matrix multiplication, element-wise product, and element-wise addition, respectively. Biases are omitted for simplicity.

*4.2.1 Hard-MGRU.* Hard-MGRU models different behaviors with different parameters and transfers information between adjacent interaction behaviors with hidden states of RNN. The structure of Hard-MGRU is shown in Figure 6(a).

For behavior types, items, and categories, we use their IDs as input of our MGRU. We first convert one-hot vectors of items and categories into embeddings of low dimensions with the following equations:

$$p_t = E^I x_t, \tag{3}$$

$$q_t = E^C c_t, \tag{4}$$

where $E^I \in \mathbb{R}^{D\times M}$ and $E^C \in \mathbb{R}^{D\times L}$ are corresponding embedding matrices. $M$ and $L$ are the number of items and categories, respectively, and $D$ is the size of embeddings. For simplicity, items and categories have the same embedding size. $x_t$ and $c_t$ are the one-hot vectors of items and categories, respectively. Item-embedding $p_t$ and category-embedding $q_t$ will be concatenated as input of our MGRU cells.

Next, Hard-MGRU will choose parameters for each interaction record according to its behavior type with equations: $W_{b_t}^r = \mathcal{W}^r b_t$, $W_{b_t}^z = \mathcal{W}^z b_t$, $W_{b_t}^h = \mathcal{W}^h b_t$. $\mathcal{W}^r$, $\mathcal{W}^z$, and $\mathcal{W}^h$ are parameters of all kinds of behaviors for reset gate, update gate, and candidate hidden state generation, respectively. Their sizes are $R \times (D * 2D)$ where $R$ is the number of behavior types. Each column of them denotes parameters of one kind of behavior. $W_{b_t}^r$, $W_{b_t}^r$, and $W_{b_t}^r$ are selected parameters according to behavior type $b_t$. They are reshaped to weight matrices of MGRU cells and the weight matrices belong to $\mathbb{R}^{D\times 2D}$. For bias parameters, they have similar process.

We use $h_t$ to denote the hidden state of the $t$th step of MGRU. $h_t$ is produced based on previous hidden state $h_{t-1}$ and current input $p_t$ and $q_t$. The equations are as follows:

$$r_t = \sigma\left(W_{b_t}^r [p_t, q_t] + b_{b_t}^r\right), \tag{5}$$

$$z_t = \sigma\left(W_{b_t}^z [p_t, q_t] + b_{b_t}^z\right), \tag{6}$$

$$\tilde{h}_t = \tanh\left(W_{b_t}^{\overline{h}} [r_t * h_{t-1}, p_t]\right), \tag{7}$$

$$\hat{h}_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \tag{8}$$

$$h_t = \frac{\hat{h}_t}{\|\hat{h}_t\|}, \tag{9}$$

where $r_t$ and $z_t$ are reset gate and update gate, respectively. $W_{b_t}$s and $b_{b_t}$s are weight matrices and biases of behavior $b_t$, respectively. $\sigma$ denotes the sigmoid function. $p_t$ and $q_t$ are concatenated as input. The last equation is for normalizing hidden states. It should be noted that our Hard-GRU is different from traditional GRU in two aspects: (1) The parameters in our MGRU cell are conditioned on behavior types and (2) Our MGRU cell has one normalization equation before outputting hidden states, which is proved effective in our experiments. What is more, we only modify the GRU cell, and other popular RNN cells like LSTM can also be modified. We leave it to be studied in the future.

Finally, our MGRU network will output a sequence of hidden states, i.e., $\{h_1, h_2, \ldots, h_T\}$. Considering that we adopt several RNNs to capture multiple coexisting intentions of users, we use $\{h_1^j, h_2^j, \ldots, h_T^j\}$ to denote hidden states of the $j$th RNN. Different RNNs have different parameters and use reset gate $r_t$ and update gate $z_t$ to decide whether the information of current behavior type should be learned by them.

*4.2.2 Soft-MGRU.* Hard-MGRU learns parameters for each kind of behavior which will cause a large space complexity. Therefore, we further propose Soft-MGRU to take advantage of gates in RNN cells to determine the importance of different behaviors and all behavior types share parameters in cells. The structure of Soft-MGRU is shown in Figure 6(b). It should be noticed that behavior types are utilized only in the computation of gates as they only determine how much information should be learned from current interaction records and contain little information about user intentions.

The input of Soft-MGRU includes behavior types, items, and categories and is the same as that of Hard-MGRU. It should be noted that, in Soft-MGRU, behavior types are also converted to embeddings of low dimensions with the following equation:

$$a_t = E^B b_t, \tag{10}$$

where $b_t$ is the one-hot vector of behavior types and $E^B \in \mathbb{R}^{D\times R}$ is the embedding matrix.

Next, the embeddings are sent to Soft-MGRU cells. For simplicity, we only introduce one cell in the multiple RNNs as they share the identical structures. The equations of Soft-MGRU are as follows:

$$r_t = \sigma(W^r[p_t, q_t, a_t] + b^r), \tag{11}$$

$$z_t = \sigma(W^z[p_t, q_t, a_t] + b^z), \tag{12}$$

$$\tilde{h}_t = \tanh(W^{\overline{h}}[r_t * h_{t-1}, p_t, q_t]), \tag{13}$$

$$\hat{h}_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \tag{14}$$

$$h_t = \frac{\hat{h}_t}{||\hat{h}_t||}, \tag{15}$$

where weight matrices $W^r$, $W^z$, and $W^{\overline{h}}$ are of the same size which is $\mathbb{R}^{D\times 3D}$. All bias $b$s are scalars. At last, Soft-MGRU outputs a sequence of hidden states, i.e., $\{h_1^j, h_2^j, \ldots, h_T^j\}$ for RNN $j$.

The difference between Hard-MGRU and Soft-MGRU lies in two aspects: (1) The parameters of Hard-MGRU are conditioned on behavior types while Soft-MGRU shares parameters for all kinds of behaviors and (2) Soft-MGRU utilizes behavior embeddings in the reset gate and update gate while Hard-MGRU does not include them in cells.

## 4.3 Multiple Tasks with MGRU

We propose an MGRU network to encode item dependency in behavior sequences and represent user intentions with hidden states. Now, we can take advantage of these hidden states to finish various tasks with different decoders. As mentioned above, we consider three tasks in this article and two of them are as auxiliary tasks.

*4.3.1 Task 1: Next Item and Category Prediction.* The first task is to predict the next item and its category that a user will interact with for all kinds of behaviors. It can be divided into two subtasks including item prediction and category prediction. In previous studies, categories are usually only used as input. The reasons for using category prediction as one subtask lie in three aspects. *First, category prediction can narrow down the prediction space and solve data sparsity problem to a certain extent.* Because the number of categories is much less than the number of items and category prediction will be much easier than item prediction. *Second, item category can reveal user coarse-grained intentions.* When users buy products, they will search for them by categories that meet their needs. *Third, item category can be seen as regularization for the representations of items.* Items of the same categories will be mapped to near points in a latent space which retains semantic information and reduces the risk of overfitting.

Considering that it is the $t$th step in a sequence, we aim to predict IDs of items and category of the $t+1$ step. In the encoder, an RNN $j$ will provide a hidden state $h_t^j$. We need to merge these hidden state $\{h_t^1, \dots, h_t^j, \dots, h_t^J\}$ from all $J$ MGRUs. As we do not know which kind of action a user will take and which intention a user will show in the next behavior, we calculate the average values of these hidden states with Equation (16).

$$h_t = mean(\{h_t^j | j = 1 \dots J\}). \tag{16}$$

Next, we predict the item and category with the following softmax functions:

$$\hat{y}_t^I = softmax(W^I h_t), \tag{17}$$
$$\hat{y}_t^C = softmax(W^C h_t), \tag{18}$$

where $W^I \in \mathbb{R}^{M \times D}$ and $W^C \in \mathbb{R}^{L \times D}$ are the parameter matrices of items and categories, respectively. $\hat{y}_t^I \in \mathbb{R}^M$ and $\hat{y}_t^C \in \mathbb{R}^L$ are the predicted probabilities of next items and categories.

*4.3.2 Task 2: Sequence Comparison.* The second task is that, given two sequences (see the preliminary section), we judge (1) whether two sequences belong to the same user and (2) whether they are adjacent sequences. Therefore, we have two binary classification subtasks. For this task, we are inspired by BERT [6] which learns word embeddings with the task of predicting next sentence.

Different from item prediction task, we only utilize the last hidden states of MGRU, i.e., $\{h_T^j | j = 1, \dots, J\}$, as representations of one sequence. We merge the last hidden states from multiple MGRU networks with mean function. The merged latent vector is denoted by $h_T$. We utilize MGRU twice and encode two sequences into two hidden states which are denoted by $h_T$ and $h'_T$, respectively. Next, we employ a fully connected neural network to generate classification results:

$$\hat{y}^S = f(h_T, h'_T), \tag{19}$$

where $f$ denotes a fully connected neural network, $\hat{y}^S \in \mathbb{R}^2$ is a vector, and the range of values in it is between 0 and 1.

*4.3.3 Task 3: Item Recommendation.* The above two tasks are auxiliary tasks. Our main task is to recommend the next new item that a user will purchase. It should be noticed that Task 1 is designed for predicting next behavior of all types. But Task 3 is used for recommending next purchased item. Similar to Task 2, we first merge the last hidden states from MGRU. Instead of using mean function, we propose an attention network to select intentions according to candidate items. The attention layer will enable each RNN to learn different user intentions. The equations of attention network are as follows:

$$\beta_j = g\left(h_T^j, e_i\right), \tag{20}$$

$$\alpha_j = \frac{\beta_j}{\sum_k \beta_k}, \tag{21}$$

$$h = \sum \alpha_j h_T^j, \tag{22}$$

where $g(\cdot)$ is a function to calculate the similarity between two vectors and can be inner product or a fully connected neural network. $e_i$ is the embedding of item $i$ and $e_i = W^I x_i$ which shares parameters with prediction tasks. $h$ is the merged latent vector which indicates users' intention.

Finally, we can calculate users' preference score $y^r$ on item $i$ with inner product:

$$\hat{y}_{u,i}^r = h^T e_i. \tag{23}$$

If user $u$ bought item $i$, $y_{u,i}^r = 1$ otherwise $y_{u,i}^r = 0$. The inner product can also be replaced by a more complex interaction function like fully-connected neural networks [10].

## 4.4 Training

For the first task, we adopt the following loss function:

$$L_1 = -\sum_u \sum_t y_t^I log\left(\hat{y}_t^I\right) + \gamma y_t^C log\left(\hat{y}_t^C\right), \tag{24}$$

where the two parts are losses of predicted items and categories, respectively.

For the second task, we use the squared loss.

$$L_2 = -\sum_i ||\hat{y}^S - y^S||_2^2. \tag{25}$$

When preparing the training dataset, we split interaction sequences of all users according to the purchase behavior records in the same way as we did in Section 3. For each user, we first select one interaction sequence $S_i^u$ and its adjacent sequence $S_{i+1}^u$ as one training sample and set their two labels as $[1, 1]$. The two labels indicate that $S_i^u$ and $S_{i+1}^u$ are of the same user and are adjacent sequences, respectively. Next, we randomly sample one nonadjacent sequence $S_j^u$, and combine it with $S_i^u$ as another training instance whose labels are set to $[1, 0]$. Then, we randomly select another user and sample one interaction sequence $S_k^{u'}$. The sampled sequence pair $S_k^{u'}$ and $S_i^u$ is as the third kinds of training instance and its two labels are $[0, 0]$. The two labels indicate that the sampled two sequences are from different users. The ratio of the above three kinds of training instances is set to $1 : 2 : 3$ to construct a balanced training dataset in our experiments.

We use cross entropy as the loss function for recommendation task.

$$L_3 = -\sum_{y^r \in \mathcal{Y}} y^r log\sigma(y^r) + (1 - y^r)log(1 - \sigma(y^r)), \tag{26}$$

where $\mathcal{Y} = \mathcal{Y}^+ \cup \mathcal{Y}^-$ is a collection of positive instances and sampled negative instances. For each positive instance, we will sample four items that the user has not interacted with as negative instances.

At last, we combine the losses from different tasks with weights.

$$L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3, \tag{27}$$

where $\lambda_i$ is weight of task $i$. We adopt mini-batch gradient decent algorithm and adam optimizer to optimize it which converges faster than other optimizers [14].

## 5 EXPERIMENTS

In this section, we conduct extensive experiments on two real-world datasets to answer the following research questions:

- *RQ1.* How does our proposed model perform as compared with state-of-the-art recommendation methods?
- *RQ2.* Are the auxiliary behaviors helpful for improving the recommendation performance of the target behavior?
- *RQ3.* Are the designed various tasks useful for the recommendation task?
- *RQ4.* Can our models learn users' intentions and improve recommendation performance with them?
- *RQ5.* Can our proposed model help address the data sparsity problem with mined coarse-grained intentions?

### 5.1 Experimental Settings

*5.1.1 Datasets and Preprocessing.* We evaluate our models with two real-world e-commerce datasets i.e., Taobao and Retailrocket. Since the statistics of the datasets have been presented, we only introduce the preprocessing steps to reduce the sparsity. We filter out users and items in Taobao datasets having fewer than 10 and 20 interaction records, respectively. For Retailrocket, we keep users and items with more than 5 and 10 records, respectively. In addition, each user should have at least one purchase record. The Retailrocket dataset consists of hierarchical (tree-like) category information. Thus, each item may belong to several categories of different levels. We only use the category information of the lowest level in our experiments. We set the maximum length of sequences to be 20. For Taobao dataset, the average length of processed sequences is about 19. For Retailrocket dataset, it is about 17.

*5.1.2 Evaluation Methodology.* We adopt leave-one-out performance validation which is widely used in previous research [7, 10]. Specifically, we leave the last purchased item as test dataset and use the rest interaction records as the training dataset. As our task is recommending next new items which will be bought by users, we delete records of the test items in the training dataset for each user. In the test dataset, we couple each instance with 99 items that the user has never interacted with as negative instances. For each user, the 100 items are ranked with the predicted preference scores from our model. The top-ranked items are treated as recommendation results. We adopt two popular evaluation metrics, i.e., HR@K (Hit ratio) [10] and NDCG@K (Normalized Discounted Cumulative Gain) [7]. $K$ is the number of recommended items. The details of the two metrics are as follows:

- *HR@K.* A recall-based measure, i.e., $HR@K = \frac{\#hits@K}{|U|}$ where $\#hits@K$ is the number of hits and $|U|$ denotes the number of users in the test dataset. It can indicate whether the test item is in the list of top-K recommended items.
- *NDCG@K.* A ranking-based measure. For each user, we compute $NDCG@K = Z_k \sum_{i=1}^{K} \frac{2^{rel_i}-1}{log_2(i+1)}$ which assigns higher scores to items with top ranks. $rel_i$ is the graded relevance value of the item at position $i$ and $Z_K$ is the normalization. In our experiment, we set $rel_i \in \{1, 0\}$, which depends on whether $i$ is the purchased item. We report the average $NDCG@K$ over all users.

*5.1.3 Baselines.* We compare our proposed model with the following baselines:

- *BPR* [24]. BPR is a matrix-factorization-based method and lets positive instance have a higher preference score than negative ones of the same user with a pairwise learning framework.

- *GRU* [12]. This work adopts GRU to model item dependence based on interaction sequences. It is designed for single-behavior data.
- *CMF* [34]. This method factorizes matrices of multiple behaviors simultaneously.
- *MC-BPR* [18]. Multi-Channel BPR adopts different sampling rules for records of different behaviors.
- *VALS* [7]. This method extends matrix factorization methods to model pairwise ranking relations among purchased, viewed, and non-viewed interactions.
- *NMTR* [9]. NMTR accounts for the cascading relationship among different behaviors with a neural network and shares embeddings of users and items among different behaviors.
- *AIR* [4]. AIR consists of two steps. First, it predicts user intention as an action-category tuple to discover category-wise sequential patterns and to capture effects of various actions for recommendation. Then, it proposes an intention-aware factorization machine to perform sequential recommendation based on the predicted intentions.
- *ASLI* [28]. ASLI learns item similarity from interaction histories with a self-attention network, obtains users' intent with a convolution network applied to item category sequences, and uses the learned intent to select the most related purchased items to predict the next item.
- *ATRank* [35]. ATRank models various user behaviors independently and the interactions between different behaviors are captured using the self-attention layers.
- *MCPRN* [31]. MCPRN utilizes several Recurrent Neural Networks to learn users' complex intentions and uses a purpose router layer to decide which RNN should be updated with current interaction records. The learned users' intentions from RNNs are selected with an attention layer to predict users' current preference score on a specific item.
- *BINN* [17]. BINN uses an item2vec method to generate embeddings of items and discriminately exploits user behaviors with two LSTM-based networks to learn historical preference and present motivation.
- *HGN* [20]. HGN applies a hierarchical gating neural network to learn the group-level representations of one sequence and utilizes item-item product to capture relations between relevant items in interaction sequences.

*5.1.4 Hyperparameter Settings.* We implement all methods with TensorFlow.[3] Since we have two choices of MGRUs (i.e., Hard-MGRU and Soft-MGRU), we name the respective methods as **IARS-H** and **IARS-S**. For each user, we recommend $K$ items and set $K$ to [2, 6, 10]. For each positive instance, we sample four items as negative instances to construct a training dataset. For tuning the hyperparameters of our methods and baselines, we leave the last purchased item of each user in the training dataset for evaluation. In what follows, we report the optimal settings. The weights of loss functions of the three tasks are set to $[0.5, 0.2, 1]$. And the weight $\gamma$ of category prediction is set to 0.5. The batch size of samples is 512. Since the findings are consistent across the number of latent factors, we report the results with the size of latent vectors and embeddings set to 64 only. The initial values of parameters in our model follow a Gaussian distribution with mean and variance equal to 0 and 0.1, respectively. The initial learning rate is set to 0.01. For ATRank, we use the code provided by the authors.[4] For all compared methods, we tune their hyperparameters and report the best performance. For all methods, the size of latent vectors is set to 64 and initial learning rate is 0.01. BPR samples four negative instances for each positive one. The maximum length of interaction sequences is 20 for GRU, AIR, ATRank, BINN, and HGN. For VALS, the $\gamma$ and

[3]https://www.tensorflow.org/.
[4]https://github.com/jinze1994/ATRank.

Table 3. Recommendation Performance (%) of Compared Methods

| | K = 2 | | | | K = 6 | | | | K = 10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Taobao | | Retailrocket | | Taobao | | Retailrocket | | Taobao | | Retailrocket | |
| Methods | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG |
| BPR | 7.66 | 6.59 | 10.01 | 9.10 | 16.04 | 10.14 | 17.05 | 12.03 | 21.67 | 11.89 | 21.93 | 13.53 |
| GRU | 9.89 | 8.50 | 10.30 | 9.65 | 21.13 | 13.28 | 18.27 | 12.45 | 29.71 | 16.00 | 23.23 | 14.36 |
| CMF | 11.42 | 10.15 | 10.39 | 10.28 | 20.53 | 13.99 | 21.74 | 13.63 | 26.70 | 15.92 | 31.04 | 16.54 |
| MC-BPR | 12.01 | 10.09 | 19.53 | 17.10 | 24.21 | 15.32 | 33.96 | 23.37 | 32.64 | 17.94 | 41.93 | 25.87 |
| VALS | 17.27 | 15.05 | 28.20 | 25.03 | 30.64 | 20.71 | 44.26 | 31.94 | 39.94 | 23.60 | 52.98 | 34.65 |
| NMTR | 12.24 | 10.73 | 26.80 | 23.87 | 25.02 | 16.13 | 41.66 | 30.22 | 33.27 | 18.71 | 50.41 | 32.95 |
| AIR | 23.63 | 19.08 | 34.90 | 33.01 | 35.87 | 25.90 | 50.56 | 40.05 | 43.69 | 28.93 | 60.13 | 42.15 |
| ASLI | 21.86 | 20.19 | 41.65 | 38.62 | 36.26 | 27.54 | 55.27 | 44.60 | 44.24 | 30.27 | 64.23 | 46.98 |
| ATRank | 25.88 | 22.75 | 36.86 | 35.73 | 37.53 | 28.57 | 53.63 | 42.46 | 44.59 | 31.11 | 64.13 | 45.25 |
| MCPRN | 27.53 | 25.64 | 30.71 | 29.44 | 39.68 | 31.14 | 49.69 | 38.28 | 46.21 | 33.28 | 61.06 | 42.07 |
| BINN | 28.71 | 26.58 | 35.64 | 34.18 | 39.45 | 32.72 | 52.28 | 41.51 | 45.99 | 35.62 | 62.31 | 43.70 |
| HGN | 25.52 | 23.81 | 38.27 | 34.61 | 38.98 | 29.58 | 55.74 | 42.15 | 46.55 | 32.10 | 63.05 | 44.43 |
| IARS-H | 32.61 | 32.52 | 43.14 | 40.46 | 43.17 | 36.56 | 57.19 | 46.46 | 49.39 | 38.15 | 65.17 | 48.95 |
| IARS-S | **36.10** | **33.64** | **47.70** | **44.86** | **45.12** | **37.54** | **60.51** | **50.39** | **50.43** | **39.20** | **67.16** | **52.48** |

$\lambda$ is set to 0.3 and 0.2, respectively. NMTR sets weights of four behavior types to $\lambda = [0.5, 1, 1, 2]$. For ASLI, different from the original model in [28], we not only utilize category embeddings in the TCN layer, but also add them to the self-attention layer. With such settings, ASLI has better results. For ATRank, the number of latent semantic spaces is set to 8. For MCPRN, the number of channels is set to 4 which is the same as ours. MCPRN does not consider the item category information. For a fair comparison between MCPRN and our methods, we concatenate item embeddings and category embeddings as the input of MCPRN. MCPRN can not deal with behaviors of multiple types. We add behavior embeddings to item embeddings for each interaction record. For HGN, we use the original implementation code provided by its authors.[5] The initial learning rate and the regularization parameter are set to 0.001 and 0.001, respectively. The embedding size is set to 64. As HGN is not designed to deal with various behaviors, we use records of all behavior types to train HGN and neglect their behavior types. Furthermore, we concatenate category embeddings with item embeddings as the input of HGN to process item category information.

## 5.2 Experimental Results

*5.2.1 Performance Comparison (RQ1).* To demonstrate the significance of our models, we first compare IARS with other methods. The results are shown in Table 3. We test all methods using HR@K and NDCG@K with $K$ set to [2, 6, 10]. From overall views, our IARS models significantly outperform all compared methods on both datasets.

BPR and GRU only utilize the data of purchase behaviors while other methods use records of all behaviors. We can see that models using all behaviors perform significantly better than BPR and GRU. GRU as one RNN-based method considers the dependency between items and has better performance than BPR. CMF method performs worse than other baselines utilizing all behaviors. It is because CMF learns several user embeddings for different behaviors which may lack training data for purchase behavior. For the rest methods, they share embeddings of users and items for all kinds of behaviors. VALS models the relative preference orders among different

[5]https://github.com/allenjack/HGN.

behaviors and achieves better performance than MC-BPR and NMTR. Unlike the above methods using only item IDs, AIR, ATRank, and BINN utilize both item IDs and categories as input and perform significantly better than other baselines. Because item categories can reveal user intentions. AIR has two steps and is not an end-to-end model. As a result, it fails to outperform ATRank and BINN. ATRank leverages self-attention to model the item dependency. But ATRank treats historical records independently and can not learn the sequential patterns among them. Similar to ATRank, ASLI also utilizes self-attention to learn item dependency. But ATRank adopts multi-head attention and ASLI has only one attention head. Therefore, ATRank can learn more complex dependency between items and performs better than ASLI in Taobao Dataset. As Retailrocket dataset is very sparse and ATRank has more parameters which may be not trained well, ATRank fails to beat ASLI. MCPRN utilizes a mixture-channel purpose routing network to learn user intentions with several RNNs and achieves good performance. But the routing delta function is not continuous and may make it hard to train the model. BINN treats purchases as the target behavior and the rest as auxiliary behaviors. It leverages two LSTMs to process the two kinds of behaviors and learn users' historical preferences and present motivations, and achieves good performance. HGN applies a hierarchical gating network to capture feature-level and group-level dependency and learns dependencies between nonadjacent items. It outperforms most baselines. HGN has better performance than BINN on Retailrocket, but fails to outperform it on Taobao dataset. The reason may be that HGN is an attention-based method and is better at processing sparse datasets (e.g., Retailrocket) than RNN-based methods like BINN.

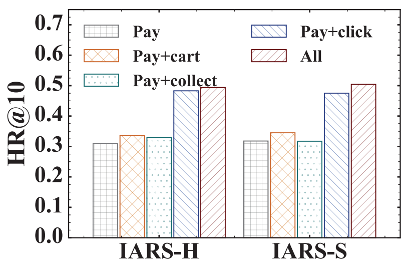
Our models perform better than all baselines, especially when $K$ is small which proves that our models have better ranking abilities. It is because our models can learn user intentions from several aspects. First, our models utilize behaviors of multiple types and auxiliary behaviors can indicate user intentions. Second, we design various tasks to reveal user intentions in behavior sequences. We not only take item categories as input, but also predict next item category as output which makes the models learn fundamental category-level intentions of users. Third, our approach model multiple coexisting intentions by using several RNNs and attention layers, thus they have stronger representation abilities than baselines. AIR, ATRank, and BINN also utilize heterogeneous behaviors and learn user intentions to some extent. AIR tries to predict item categories and behavior types in the next interaction record as user intentions in its first step. But AIR is not an end-to-end model. ATRank utilizes the self-attention layers and learns user interests in several semantic spaces. However, it fails to model the sequential patterns in behaviors. BINN considers long-term and short-term user interests, but can not distinguish various intentions in recent behaviors. Among our proposed models, Soft-MGRU seems to perform a little better than Hard-MGRU. We think it is because that Hard-MGRU does not share parameters for different behaviors in RNN cells. The parameters of purchase behaviors may not be trained very well due to the data sparsity problem.

To further prove that our proposed approaches have better performance than baselines, we adopt another evaluation method and other three metrics which are the same as that in [20] to evaluate our approaches. For each user, we leverage 70% of the interaction records as the training set and the next 10% of records for tuning hyper-parameters. The remaining 20% of records generated by users are as the test set. For each user, we remove the records of items in the test set from training set to ensure that the items did not appear in the user's history sequences. We utilize our approaches and baselines to rank all items and calculate precision, recall and NDCG with the top 10 recommended items for each user. We report the average values of the three metrics.

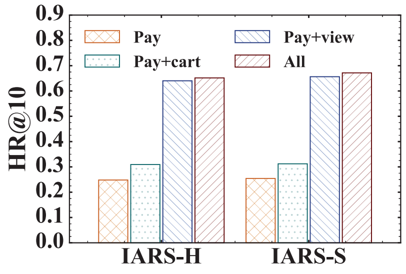
The experimental results of several best baselines and our approaches are shown in Table 4. We can see that our proposed approaches still outperform all baselines on both datasets. This experiment further proves the effectiveness of our approaches.

Table 4. Recommendation Performance with Non-Sampled Evaluation Metrics (k = 10)

| | Taobao | | | Retailrocket | | |
|---|---|---|---|---|---|---|
| Methods | Precision | Recall | NDCG | Precision | Recall | NDCG |
| ATRank | 0.0234 | 0.0250 | 0.0318 | 0.0209 | 0.0470 | 0.0403 |
| MCPRN | 0.0256 | 0.0281 | 0.0315 | 0.0194 | 0.0459 | 0.0367 |
| BINN | 0.0279 | 0.0317 | 0.0345 | 0.0211 | 0.0566 | 0.0497 |
| HGN | 0.0276 | 0.0286 | 0.0387 | 0.0201 | 0.0647 | 0.0540 |
| IARS-H | 0.0301 | 0.0341 | 0.0363 | 0.0239 | 0.0826 | 0.0565 |
| IARS-S | **0.0349** | **0.0388** | **0.0401** | **0.0253** | **0.0830** | **0.0601** |



(a) Taobao dataset.



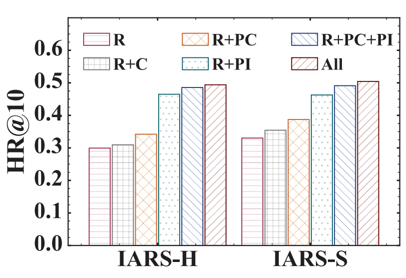
(b) Retailrocket dataset.

Fig. 7. Effect of auxiliary behaviors.

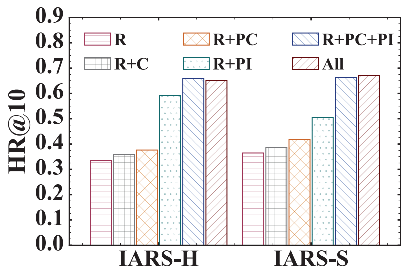
5.2.2 *Impact of Multiple Behaviors (RQ2).* As we exploit heterogeneous behaviors to learn user intentions and assist the recommendation task, we investigate the influence of various behaviors on recommendation performance. The results of our models with different kinds of behaviors are shown in Figure 7. We can see that with only purchase behaviors, the performance of our models drops a lot. But it is still better than that of BPR and GRU in Table 3 which proves the effectiveness of normalization in our cells. Among all auxiliary behaviors, click and view are the most important behaviors for improving performance as they have more records than other behaviors and frequent-occurring item categories can reveal users' category-level intentions.

5.2.3 *Impact of Auxiliary Tasks (RQ3).* To clarify the effectiveness of auxiliary tasks, we show the results of our models with different tasks in Figure 8. As the next-item prediction task includes item ID prediction and category prediction, we split it into two tasks. In the figure, our models with all tasks have the best performance. The sequence comparison task improves the performance of the models with only recommendation task. However, it seems that the improvement of adding prediction tasks is much larger. It may be because item IDs and categories contain more intention information than the labels (only 0 and 1) in the comparison task. Item ID and category prediction tasks both play important roles in improving recommendation performance.

5.2.4 *Visualization of Item Embeddings (RQ4).* We visualize the embeddings of items to show whether our models learn the semantic information of item category. However, the size of item embeddings is 64 which can not be visualized. We use the T-SNE [21] to reduce the dimension of embeddings and show the embeddings in Figure 9. Each item point is colored according to its category. For comparison, we also show the item embeddings from BINN which is the best baselines

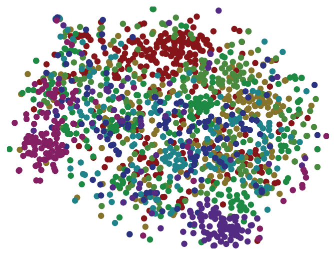


(a) Taobao dataset.

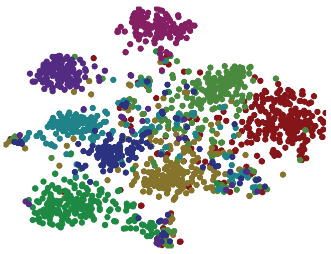


(b) Retailrocket dataset.

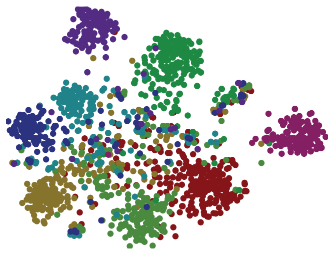
Fig. 8. Effect of auxiliary tasks. R and C denote recommendation task and sequence comparison task, respectively. PC and PI denote item ID and category prediction tasks.



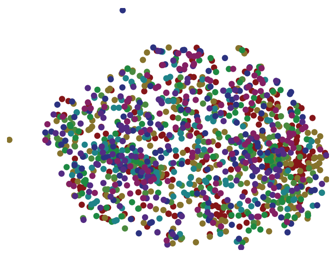
(a) BINN (Taobao dataset).



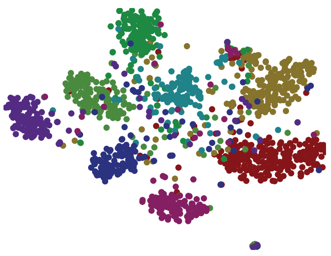
(b) IARS-H (Taobao dataset).



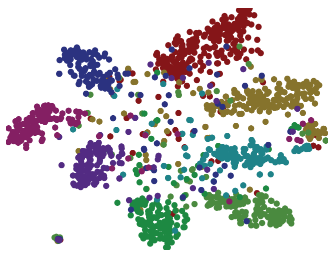
(c) IARS-S (Taobao dataset).



(d) BINN (Retail dataset).



(e) IARS-H (Retail dataset).

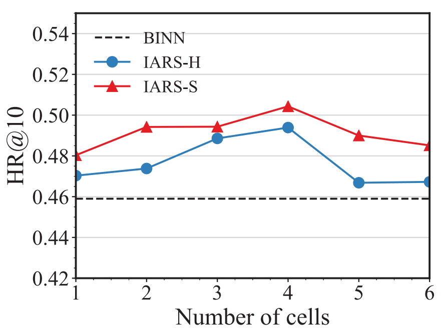


(f) IARS-S (Retail dataset).

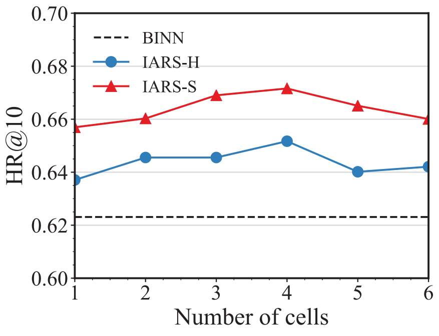
Fig. 9. T-SNE embedding for item vectors produced by BINN, IARS-H and IARS-S. Items of different categories are denoted by points of different colors.

in our experiments. We can see that for our models, items of the same categories cluster together. Such a phenomenon is not very clear for BINN. It proves that our models learn the category information which is quite important for mining user intentions. It seems that Hard-MGRU is better at bringing together items of the same categories than Soft-MGRU.

*5.2.5 Effect of the Number of MGRU Cells (RQ4).* We study the effectiveness of utilizing several RNNs to model users' complex intentions. The performance of our models with a different number of MGRU cells is shown in Figure 10. From the figure, we can see that the performance of our models grows at first. With more RNNs, our approach can model more complex user intention which improves the performance. It seems that when the number of cells is four, our models achieve the best performance for both datasets which proves the effectiveness of utilizing multiple RNNs to



(a) Taobao dataset.



(b) Retailrocket dataset.

Fig. 10. Effect of the number of MGRU cells.

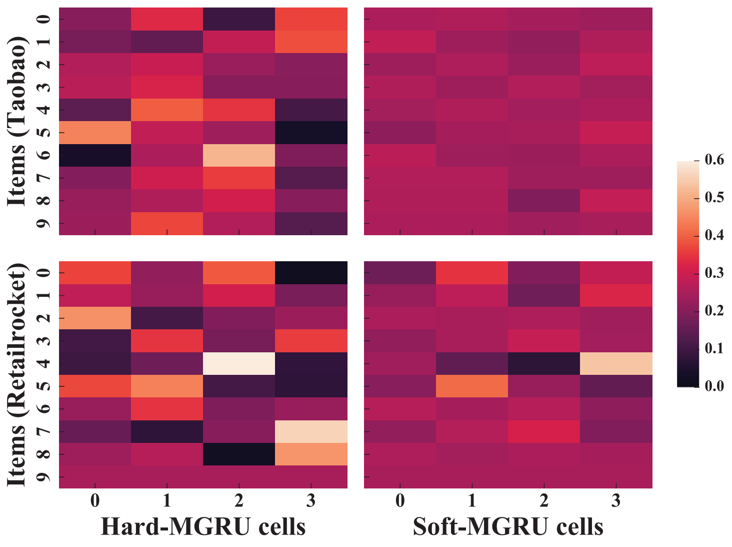
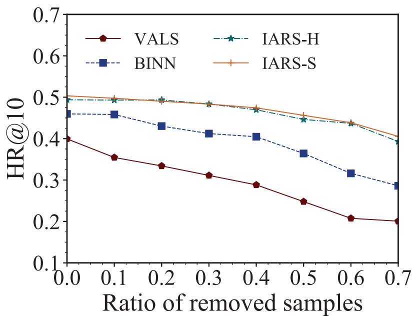


Fig. 11. Visualization of attention scores on MGRU cells.

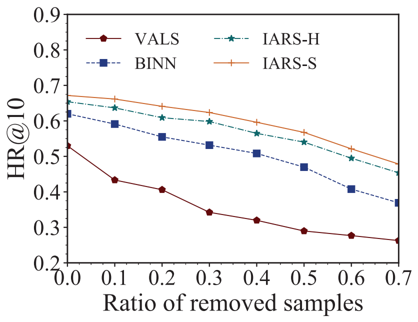
model users' complex intentions. And then it drops a little as the number of cells continues to grow which may be caused by overfitting.

5.2.6 *Visualization of Attention (RQ4).* We select ten samples and visualize the attention scores of items in Figure 11 to show whether different MGRUs can learn different user intentions. In the figure, the first row and the second row show attention scores of Taobao dataset and Retailrocket dataset, respectively. The figures in the two columns are results of Hard-MGRU and Soft-MGRU, respectively. The figure shows that attention scores of MGRUs vary for items. The variance of attention scores of Hard-MGRU seems larger than that of Soft-MGRU. It may be because that Hard-MGRUs have different parameters for complex interaction behaviors. It proves that Hard-MGRU is more suitable to distinguish various intentions.

5.2.7 *Impact of Data Sparsity (RQ5).* We further study the impact of data sparsity on the performance of our proposed methods. As our models can learn users' intentions, they should have better performance than baselines. We randomly remove some samples from the training dataset. The ratio of removed training samples ranges from 0 to 0.7. We choose two baselines, i.e., VALS and



(a) Taobao dataset.



(b) Retailrocket dataset.

Fig. 12. Effect of data sparsity.

BINN, to be compared with our models. Because BINN performs the best among all baselines and can leverage categories of items to reveal users' coarse-grained intentions to some extent. VALS utilizes behaviors of multiple types, but cannot use item category information. In other words, VALS is weak in learning users' intentions.

The experimental results in terms of HR@10 are shown in Figure 12. We can observe that our models outperform the other two methods. As the ratio of removed samples rises, the performances of all compared models drop. But the decreasing rate of our model is smaller than that of other methods. The phenomenon is very clear on the Taobao dataset. Furthermore, the performance of BINN drops slower than VALS, especially on the Retailrocket dataset. It is because that our models and BINN utilize item category information to reveal users' coarse-grained intentions. When the training dataset is sparse, they can achieve acceptable performance. In addition, our models try to predict interacted categories in the future and implicitly learn the users' intention information. As a result, our model achieves better performance than BINN which uses categories only in input.

## 6 CONCLUSION

In this article, we propose to exploit behaviors of multiple types for next-item recommendation. Through empirical studies, we find that users have multiple coexisting category-level intentions which evolve with time. To process the multi-behavior data and mine the evolving user intentions, we offer two kinds of RNN cells, i.e., Hard-MGRU and Soft-MGRU, and design two auxiliary tasks including next item category prediction and sequence comparison. Considering the purchase intentions are complex, we propose to leverage multiple MGRUs and an attention layer to learn these coexisting intentions. Extensive experiments on two real-world datasets show that our models outperform several state-of-the-art methods.

## REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[2] Yoshua Bengio and Olivier Delalleau. 2011. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*. 18–36.

[3] Hung-Hsuan Chen. 2018. Behavior2Vec: Generating distributed representations of users' behaviors on products for recommender systems. *TKDD* 12, 4 (2018), 43.

[4] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. AIR: Attentional intention-aware recommender systems. In *ICDE*. 304–315.

[5] Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556* (2017).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving implicit recommender systems with view data. In *IJCAI*. 3343–3349.

[8] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*. 278–288.

[9] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *ICDE*. 1554–1557.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[11] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

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

[13] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. 2018. Improving entity recommendation with search log and multi-task learning. In *IJCAI*. 4107–4114.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.

[16] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*. 173–182.

[17] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *SIGKDD*. 1734–1743.

[18] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 361–364.

[19] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2019. Top-N recommendation with multi-channel positive feedback using factorization machines. *ACM Trans. Inf. Syst.* 37, 2 (2019), 15:1–15:23.

[20] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *SIGKDD*. 825–833.

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[22] 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 *SIGKDD*. 596–605.

[23] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[25] Michael L. Seltzer and Jasha Droppo. 2013. Multi-task learning in deep neural networks for improved phoneme recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 6965–6969.

[26] Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *SIGKDD*. 650–658.

[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. In *CIKM*. 1441–1450.

[28] Md Mehrab Tanjim, Congzhe Su, Ethan Benjamin, Diane Hu, Liangjie Hong, and Julian McAuley. 2020. Attentive sequential models of latent intent for next item recommendation. In *WWW*. 2528–2534.

[29] Mengting Wan and Julian J. McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys*. 86–94.

[30] 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*. 2000–2010.

[31] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Longbing Cao. 2019. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *IJCAI*. 3771–3777.

[32] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. 2019. Next item recommendation with self-attentive metric learning. In *AAAI Workshop on Recommender Systems and Natural Language Processing*, Vol. 9.

[33] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*. 94–108.

[34] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H. Chi. 2015. Improving user topic interest profiles by behavior factorization. In *WWW*. 1406–1416.

[35] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *AAAI*. 4564–4571.

[36] Yu Zhu, Yu Gong, Qingwen Liu, Yingcai Ma, Wenwu Ou, Junxiong Zhu, Beidou Wang, Ziyu Guan, and Deng Cai. 2019. Query-based interactive recommendation by meta-path and adapted attention-GRU. In *CIKM*. 2585–2593.

[37] Yu Zhu, Junxiong Zhu, Jie Hou, Yongliang Li, Beidou Wang, Ziyu Guan, and Deng Cai. 2018. A brand-level ranking system with the customized attention-GRU model. In *IJCAI*. 3947–3953.

Received May 2020; revised October 2020; accepted December 2020