Modeling Multi-aspect Preferences and Intents for Multi-behavioral Sequential Recommendation

Haobing Liu[a,b], Jianyu Ding[b], Yanmin Zhu[b,*], Feilong Tang[b], Jiadi Yu[b], Ruobing Jiang[a], Zhongwen Guo[a]

[a]Department of Computer Science and Technology, Ocean University of China
[b]Department of Computer Science and Engineering, Shanghai Jiao Tong University

Abstract

Multi-behavioral sequential recommendation has recently attracted increasing attention. However, existing methods suffer from two major limitations. Firstly, user preferences and intents can be described in fine-grained detail from multiple perspectives; yet, these methods fail to capture their multi-aspect nature. Secondly, user behaviors may contain noises, and most existing methods could not effectively deal with noises. In this paper, we present an attentive recurrent model with multiple projections to capture **M**ulti-**A**spect preferences and **INT**ents (MAINT in short). To extract multi-aspect preferences from target behaviors, we propose a multi-aspect projection mechanism for generating multiple preference representations from multiple aspects. To extract multi-aspect intents from multi-typed behaviors, we propose a behavior-enhanced LSTM and a multi-aspect refinement attention mechanism. The attention mechanism can filter out noises and generate multiple intent representations from different aspects. To adaptively fuse user preferences and intents, we propose a multi-aspect gated fusion mechanism. Extensive experiments conducted on real-world datasets have demonstrated the effectiveness of our model.

Keywords: Multi-aspect Preferences, Multi-aspect Intents, Multi-behavioral

[*]Corresponding author
Email addresses: haobingliu@ouc.edu.cn (Haobing Liu), dingjianyu@sjtu.edu.cn (Jianyu Ding), yzhu@sjtu.edu.cn (Yanmin Zhu), tang-fl@cs.sjtu.edu.cn (Feilong Tang), jiadiyu@sjtu.edu.cn (Jiadi Yu), jrb@ouc.edu.cn (Ruobing Jiang), guozhw@ouc.edu.cn (Zhongwen Guo)

1. Introduction

With the rapid growth of the amount of information, recommender systems seeking to predict items that a user may have an interest in have become fundamental for helping users overcome information overload. Since user preferences may change over time, sequential recommendation has been studied, which can model sequence-related patterns (such as sequential patterns, co-occurrence patterns, and distance patterns) in user-item interactions [1].

Recently, various types of user interactions have increasingly been collected [2, 3, 4]. For example, an e-commerce website may collect click, add-to-cart, and buy behaviors. However, most existing recommender models only consider *the target type of behavior*, which directly relates to the KPI and is believed to be the strongest signal reflecting a user's preference. As such, *support types of behaviors* are often neglected. This is problematic because support types of behaviors can provide vital clues about a user's intent. Both user preference (what users love) and intent (what users currently want) are critical factors that influence whether or not a user is interested in an item [5].

To leverage multi-typed behaviors, various existing methods have been proposed and they can be divided into two categories. The first category mainly involves extending the Matrix Factorization (MF) algorithm [6], Bayesian Personalized Ranking (BPR) algorithm [7], or using Multi-Layer Perceptron (MLP) to seek novel solutions. However, we argue that this category models multi-typed behaviors from a static perspective, which neglects sequence-related patterns in user-item interactions. The second category considers both the multi-typed and sequential nature of behaviors, and can be further divided into two subcategories. The first subcategory regards all behaviors as one long sequence and designs methods that aim to handle it [8, 9]. However, this subcategory fails to capture the intrinsic patterns of each type of behavior. The second subcategory models at least two kinds of sequences to alleviate the limitations of the

Figure 1: Recommending by integrating multi-aspect preferences and intents.

first subcategory [10, 11, 12]. However, it still fails to consider multiple aspects of user preferences and intentions. Besides, most methods ignore that support types of behavior may be very noisy. If noises are not handled properly, they may harm the performance of models.

Actually, user preferences and intents are multi-aspect. In other words, user preferences and intents should be described in a fine-grained way. This can be better understood referring to Figure 1 as an example. User preferences and intents are described from two main aspects: the category-aspect and the brand-aspect. The historical buy behaviors of user 1 imply that she loves potato chips. Additionally, her click records indicate that she is shopping for Oishi products. By considering both the preferences and intents of user 1, it would be appropriate to recommend Oishi potato chips to her.

We find three key technical challenges in multi-behavioral sequential recommendation. First, we try to learn user intents from her multi-typed behavioral sequence. However, different behavioral specifics (e.g., behavior type, time interval) which provide a fine-grained description of one user interaction on one item indicate different strengths of the intents. For example, *add-to-cart* could be a stronger signal compared with *click*. Besides, the multi-typed behavioral sequence can be noisy. To support this observation, we calculate the conversion rates of different behavior types using real-world datasets. The results, shown in Table 1, reveal extremely small conversion rates of *click*, *collect*, and *view*. This indicates that noises can be introduced when utilizing support types of

Table 1: Conversion rates of different behavior types on two datasets.

Dataset	Taobao		Retailrocket	
	0.09	Click	0.16	View
Conversion Rate	0.07	Collect		
			0.76	Cart
	0.26	Cart		

behaviors to help to predict the next target behavior. Thus, a challenge lies in *how to extract users' intents from multi-typed behavioral sequences while considering the behavioral specifics and the noisy nature of the sequences.* Second, user preferences and intents are multi-aspect. Figure 1 provides an example that illustrates how user preferences and intents are reflected in the category-aspect and the brand-aspect. While explicit aspects like category and brand help explain the multi-aspect nature of preferences and intents, these aspects are implicit in the real world. Leveraging one latent vector to represent a user's preferences/intents is not enough to capture the multi-aspect nature. Thus, the second challenge is on *how to capture the multi-aspect nature of users' preferences and intents.* Third, preferences and intents have different degrees of impact on different users. Considering users who only made few purchases a long time ago but clicked many items recently, their intents would be more important compared with their preferences. Simple fusion methods (e.g., concatenation [10, 11]) may be inappropriate. Thus, *how to fuse multi-aspect preferences and intents adaptively* is challenging.

To address the aforementioned challenges, we propose a novel attentive recurrent model with multiple projections for capturing **M**ulti-**A**spect preferences and **INT**ents (MAINT in short). *To model user intents from noisy multi-typed behavioral sequences*, we design a behavior-enhanced LSTM and a refinement attention mechanism. By adding behavioral specifics in the gates of Long Short-Term Memory (LSTM) [13], LSTM can consider behavioral specifics while modeling an item sequence. The refinement attention mechanism regards the stable preference as a guider to filter out noises during intent extraction. *To capture the*

4

multi-aspect nature of preferences and intents, we propose a multi-aspect projection mechanism and leverage parallel attention mechanisms. In detail, we first project the hidden state of the LSTM modeling the target behavior sequence to multiple semantic subspaces representing multiple aspects. This results in multiple representations of hidden user preferences from different aspects. Then, we regard these preferences as guiders and perform the refinement attention mechanisms in parallel, obtaining representations of multi-aspect latent intents. *To adaptively fuse multi-aspect preferences and intents*, we develop a multi-aspect gated fusion mechanism. The mechanism can balance the impacts of preferences and intents by considering their characteristics in each aspect.

The main contributions of this work are as follows:

- We propose a novel approach for multi-behavioral sequential recommendation. To the best of our knowledge, it is the first model which can extract and fuse multi-aspect preferences and intents.

- To model multi-aspect preferences, we design a multi-aspect projection mechanism that generates multiple preference representations from different aspects.

- To model multi-aspect intents, we design a behavior-enhanced LSTM and a multi-aspect refinement attention mechanism. The LSTM captures specific while modeling an item sequence. The attention mechanism generates multiple intent representations from different aspects.

- To fuse multi-aspect preferences and intents, we propose a multi-aspect gated fusion mechanism. In each aspect, there is a gating mechanism that integrates the preferences and intents from that aspect.

2. Related Work

We classify related studies into three categories: multi-behavioral non-sequential recommendation, single-behavioral sequential recommendation, and multi-behavioral sequential recommendation as Table 2 shows.

5

Table 2: Categorization of related studies.

Category	Approaches	References
Multi-behavioral Non-sequential Recommendation	Traditional Methods	[14, 15]
	Deep Learning-based Methods	[2, 16, 17, 18, 19, 20, 21]
Single-behavioral Sequential Recommendation	Traditional Methods	[22, 23, 24]
	RNN-based Methods	[25, 26, 27, 28]
	CNN-based Methods	[29, 30]
	Transformer-based Methods	[31, 32, 33, 34]
	GNN-based Methods	[35, 36, 37]
	Other Methods	[38]
Multi-behavioral Sequential Recommendation	Methods Taking One Kind of Sequence as Input	[9, 8]
	Methods Taking At Least Two Kinds of Sequence as Input	[10, 39, 12, 40, 41]

2.1. Multi-behavioral Non-sequential Recommendation

Multi-behavioral non-sequential recommender methods are designed to capture the heterogeneity of behaviors. According to whether utilizing deep learning structures, methods can be divided into traditional methods and deep learning-based methods.

2.1.1. Traditional Methods

Traditional methods are mainly based on the MF algorithm or the BPR algorithm. For example, Zhao et al. [14] jointly factorized multiple matrices of different behavior types with sharing item-side embeddings. Loni et al. [15] proposed sampling rules considering levels of different behavior types.

2.1.2. Deep Learning-based Methods

Some researchers attempt to leverage deep neural networks. For example, NMTR [2] combines the advance of NCF [42] with multi-task learning. For each behavior type, one specific interaction function is learned. Based on multi-typed behavioral data, a user-item bipartite graph with multiple types of edges

can be constructed. Then Graph Neural Networks (GNNs) could be utilized to handle the user-item multi-behavior graph. Jin et al. [16] assigned different learnable weights to different kinds of edges with graph attention networks to model the importance of the behaviors. Similarly, Xia et al. [17] designed an aggregation mechanism which is based on the attentional neural mechanism to explicitly model the dependencies between different types. Chen et al. [18] assigned representations to behavior types (i.e., edge types). Zhang et al. [19] proposed a graph convolutional network-based method to learn user and item representations under different behavior types. Moreover, some researchers explore heterogeneous graph embedding methods [20, 21]. We argue these methods mainly ignore sequence-related patterns of behaviors.

2.2. Single-behavioral Sequential Recommendation

Most sequential recommendation methods focus on handling single-behavioral sequences. They strive to model sequential dependencies existing in user interactions [43]. We first review traditional methods. Then, we summarize Recurrent Neural Network (RNN)-based methods, Convolution Neural Network (CNN)-based methods, transformer-based methods, GNN-based methods, and other methods.

2.2.1. Traditional Methods

Traditional methods are mainly based on Markov Chains (MCs). For example, Rendle et al. [22] combined the MF with first-order MCs. He et al. [23] further integrated an item similarity model with high-order MCs.

Hosseini et al. [24] proposed a method to retrieve multi-aspect temporal similarity maps. The method aims to alleviate the sparseness issue in the user-location matrix. Additionally, they utilize the Expectation-Maximization technique to compensate for incomplete data at each temporal scale.

2.2.2. RNN-based Methods

RNNs have been widely adopted to model sequential data (e.g., sequences of words) and have shown efficient performance. Hidasi et al. [25] introduced

the Gated Recurrent Unit (GRU) [44] to model click behaviors. Zhu et al. [26] modified the gate structure of the vanilla LSTM cell. Wang et al. [27] proposed a neural network to learn multiple purposes of users. Several RNNs were used to learn different purposes and a purpose router was designed to decide which RNN should be updated with an interacted item. Check-in behaviors are spatio-temporal data. Zhao et al. [28] extended the vanilla LSTM cell with additional spatio-temporal gates, enabling the utilization of time and distance intervals. These methods cannot ignore the heterogeneity of behaviors.

2.2.3. CNN-based Methods

Besides RNNs, some researchers explore CNNs. CNNs are commonly applied to handle image data. Tang and Wang [29] regarded the embedding matrix $E \in \mathbb{R}^{L \times d}$ of L previous items as an "image". They used a vertical horizontal convolutional layer and a horizontal convolutional layer to handle the "image". Yuan et al. [30] also regard an item embedding sequences as an "image" and explored holed CNN layers. Since CNNs are suitable to capture local features, these methods cannot well capture the dependencies between two distant interactions.

2.2.4. Transformer-based Methods

Latterly, transformer [45] has been introduced into recommender systems. For example, SASRec [31] is based on a self-attention network. Zhang et al. [32] applied parallel self-attention blocks on both item sequences and feature sequences. Luo et al. [33] constructed spatio-temporal relation matrices and utilized a self-attention layer to capture the dependencies between non-adjacent POIs and non-contiguous visits. Saaki et al. [34] leveraged discrete-time modeling to exploit concepts from texts and continuous-time modeling to infer infinite behavioral patterns of users. These methods model sequential dependencies with the help of the positional encoding technique. Researchers have developed various positional encoding schemes [46]. Choosing an improper positional encoding scheme will result in incorrect time modeling.

2.2.5. GNN-based Methods

In addition, some researchers try to transform sequences into graph-structured data and utilize GNNs for sequential recommendation. For example, Wu et al. [35] transformed item sequences into item-item directed graphs and leveraged a gated graph neural network to handle the graphs. Li et al. [36] constructed graph-augmented POI sequences to capture the collaborative signals from semantically correlated POIs (i.e., POIs before and after the target POI in different sequences). They proposed a position-aware attention net to model sequential dependencies. Rao et al. [37] first constructed a spatial-temporal knowledge graph and leveraged a knowledge graph embedding method to learn embeddings of entities and relations. Next, they constructed a POI transition graph based on the embeddings. Then, they incorporated the POI transition graph into RNN-based models. We argue the transformations are not one-to-one mappings, thus they may be lossy. Besides, these methods may not effectively capture long-term dependencies.

2.2.6. Other Methods

Other deep learning structures have also been adopted in the single-behavioral sequential recommendation. For example, Ma et al. [38] specifically designed a hierarchical gating network with BPR to capture both long-term and short-term user preferences. To capture the short-term user preferences, they designed a feature gating and an instance gating for hierarchically selecting features and items.

2.3. Multi-behavioral Sequential Recommendation

For capturing the multi-typed and sequential nature of interactions, researchers have developed multi-behavioral sequential recommender models, which can be further classified into two subcategories.

2.3.1. Methods Taking One Kind of Sequence as Input

The first subcategory regards all behaviors as one long sequence. Liu et al. [9] combined RNN with log-bilinear model to handle the long sequence. To capture

properties of multiple types of behaviors, they incorporated behavior-specific matrices. Zhou et al. [8] leveraged an attention-based RNN to model sequential information of the long sequence. They learned representations for each behavior type and concatenated the representations and item representations as input. We argue that these methods cannot capture the intrinsic patterns of each type of behavior well.

2.3.2. Methods Taking At Least Two Kinds of Sequence as Input

The second subcategory models at least two kinds of sequence. For example, Li et al. [10] proposed an LSTM-based model to learn from the preference behaviors and the session behaviors. Zhou et al. [39] grouped behaviors by behavior type and leveraged a self-attention network to model the influence among different behavior types. Tanjim et al. [12] learned item similarities based on the interacted item sequence via a transformer layer and obtained the user's intent based on her actions on a particular category via a convolution layer. Xu et al. [40] utilized multiple GRUs to learn diverse intentions. It designs three tasks to improve recommendation performance. Wu et al. [41] adopted contrastive learning among different behavior types. We argue that these studies fail to consider the multi-aspect nature of both preferences and intents. Furthermore, most studies ignore that support types of behaviors can be noisy.

3. Preliminaries

3.1. Problem Formulation

The main notations utilized throughout this paper are outlined in Table 3. While users interact with recommender systems, interactions will be recorded. Given a user u, interactions of u can be defined as $X^u = \{x_1, x_2, \cdots, x_N\}$. The n-th element $x_n = (i_n, c_n, t_n, b_n)$ indicates that u performs a *behavior* of type b_n on the item i_n at the timestamp t_n. All users constitute a user set U. All items occurring in all behaviors constitute an item set I. All behavior types constitute a behavior type set B. c_n is the content information (such as category information and brand information) of item i_n.

10

Table 3: Main notations.

Notation	Description
$x = (i, c, b, t)$	one behavior on item i whose content information is c (The behavior type is b and it happened at t.)
$X^u = \{x_1, x_2, \cdots, x_N\}$	the behavior sequence belonging to user u
p, q	the embedding for item i, item content c
r, s	the embedding for behavior type b, time interval Δt
J	the number of implicit aspects
$\tilde{h}_j^S$	the projected stable preference representation in the j-th aspect
$\alpha_{j,n}$	the attention score assigned to the n-th item in the j-th aspect
$\tilde{h}_j^D$	the dynamic intent representation in the j-th aspect
β_j	the gate scores assigned to the latent intent in the j-th aspect
$\tilde{h}_j^H$	the hybrid user representation in the j-th aspect
h^F	the final user representation
$\hat{y}^I$	the predicted probabilities of items

In recommender systems, there exists a key type of behavior referred to as *target type*. On e-commerce websites, the target type is usually *buy*. For ease of illustration, we use "buy" or "target" interchangeably throughout this paper. The other types of behaviors belong to *support types*.

Problem Formulation. The problem of multi-behavioral sequential recommendation is formulated as follows:

Input: Users U, items I, behavior types B, and all users' behavioral data X.

Output: A predictive model aims to predict the item that each user is most likely to interact with under the target behavior type.

3.2. Framework Overview

Figure 2 illustrates our framework, consisting of two phases: offline and online, to recommend the top K items that each user is most likely to interact with under the target behavior type.

- **Offline Phase.** The Data Preprocessor processes User Logs in accordance

Figure 2: Framework.

with the supported input format to generate inputs for the Recommender Model. The Recommender Model is trained using the training data to obtain well-trained model parameters.

- **Online Phase.** The Data Preprocessor processes User Behavioral Data to generate inputs for the model. The model is deployed on servers and the trained model parameters are loaded. The Real-time Recommender Server ranks candidate items based on each user's historical behavior and presents the user with a sorted list of the top-ranked items.

4. Proposed Model

We next present the detailed design of our model, whose structure is presented in Figure 3. Our model has three components: 1) multi-aspect preference modeling component; 2) multi-aspect intent modeling component; 3) preference and intent fusing component. We will illustrate our model using the example from Figure 1.

Figure 3: The overview of the MAINT structure. Figure best viewed in color.

4.1. Shared Embedding

A shared embedding layer is utilized to map items, item content features, and behavior types into embedding vectors:

$$p_n = W^I i_n, \quad q_n = W^C c_n, \quad r_n = W^B b_n, \tag{1}$$

where W matrices are embedding matrices. i_n and b_n are one-hot vectors. c_n can be a one-hot or multi-hot vector. Since we only use item category information which is widely used in previous studies [12, 39], c_n is a one-hot vector in this paper. If other item content information was available, it could also be included. The time interval between x_n and x_{n+1} is calculated as $\Delta t_n = t_{n+1} - t_n$. Following previous studies [39], we divide the overall time intervals into buckets and map each one-hot bucket vector into an embedding vector s_n with W^Δ:

$$s_n = W^\Delta \text{bucketize}(\Delta t_n). \tag{2}$$

4.2. Multi-aspect Preference Modeling

Stable preferences are with fewer fluctuations and can be reflected by historical target behavior sequences. In the context of a multi-typed behavioral sequence $\{x_1, x_2, \cdots, x_N\}$ of length N, we pick up target behaviors to form a

13

target behavior sequence $\{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_M\}$ [1], $M < N$. Reviewing Figure 1, the multi-typed behavioral sequence of User 1 can be represented as $\{x_1, x_2, x_3, x_4\}$. x_1 and x_2 are target behaviors and the target behavior sequence is represented as $\{\bar{x}_1, \bar{x}_2\}$. After the embedding layer, the item and item content embeddings are sent to an LSTM layer:

$$h_m^S = \text{LSTM}([p_m, q_m], h_{m-1}^S), \quad 1 \le m \le M, \tag{3}$$

where $[,]$ represents the concatenation of vectors and h_m^S is the hidden state at the m-th step. p_m, q_m are calculated with Equation (1).

User preferences are multi-aspect. To model multi-aspect preferences, we propose a multi-aspect projection mechanism. More specifically, we project the hidden state h_M^S to multiple semantic subspaces, i.e., aspects:

$$\tilde{h}_j^S = \mathcal{P}_j(h_M^S), \quad 1 \le j \le J, \tag{4}$$

where $\mathcal{P}_j$ is the projection function for the j-th implicit aspect and $\tilde{h}_j^S$ is the projected preference representation. The rationality of the multi-aspect projection mechanism is easy to be proved. Many studies [47] have proved that different filters in a convolutional layer focus on distinct features of an image. Our projection functions are similar to filters. Each projection function focuses on automatically extracting characteristics of a specific aspect. Reviewing Figure 1, we should project user preferences to two semantic subspaces: the category-aspect and the brand-aspect. Then, we obtain user preference representations of User 1: $\tilde{h}_1^S$ and $\tilde{h}_2^S$.

The projection functions can either be linear or nonlinear. For simplicity, we present the linear projection function:

$$\tilde{h}_j^S = W_j^P h_M^S, \quad 1 \le j \le J, \tag{5}$$

where W^P are the projection matrices.

---

[1] We use $\bar{x}_\star$ to represent a target behavior.

4.3. Multi-aspect Intent Modeling

Dynamic purchase intents evolve over time and can be reflected by multi-typed behavioral sequences [12]. A common scenario is that when a user wants something, she will click some related items and then add some of these items to her cart.

Behavior type and time interval are behavioral specifics which provide a fine-grained understanding of one user interaction on one item. Although behavior specifics cannot provide any information about the item, they reflect the importance of each interaction. To consider behavioral specifics while modeling an item sequence, we regard behavioral specifics as strong signals in input, forget, and output gates. That is to say, behavioral specifics are involved in controlling item sequence modeling. This variant of LSTM is called behavior-enhanced LSTM, which is formulated as follows:

$$i_n = \sigma(W_i[p_n, q_n, r_n, s_n, h_{n-1}] + \hat{b}_i), \tag{6}$$

$$f_n = \sigma(W_f[p_n, q_n, r_n, s_n, h_{n-1}] + \hat{b}_f), \tag{7}$$

$$\hat{c}_n = f_n \odot \hat{c}_{n-1} + i_n \odot \tanh(W_{\hat{c}}[p_n, q_n, h_{n-1}] + \hat{b}_{\hat{c}}), \tag{8}$$

$$o_n = \sigma(W_o[p_n, q_n, r_n, s_n, h_{n-1}] + \hat{b}_o), \tag{9}$$

$$h_n = o_n \odot \tanh(\hat{c}_n), \tag{10}$$

where i_n, f_n and o_n are gates at the n-th step. $\hat{c}_n$ is the cell memory. W matrices and $\hat{b}$ terms are trainable parameters. σ is the element-wise sigmoid function. $\odot$ is the element-wise product. p_n and q_n are item characteristics and calculated with Equation (1). r_n and s_n are behavioral specifics and can be calculated with Equation (1) and Equation (2), respectively.

When modeling multi-typed behavioral sequences, it is important to consider the presence of noise. For example, click behaviors can be quite noisy due to accidental touches. To mitigate this issue, we regard stable preferences as guiders. In detail, we employ a refinement attention mechanism in each aspect. The attention mechanism utilizes the stable preference as a guider to guide the dynamic intent extraction from the sequence by assigning different weights. In this

Figure 4: Refinement Attention Mechanism.

way, we will learn J intent representations from J aspects. Refinement attention mechanisms constitute the multi-aspect refinement attention mechanism, which is inspired by the success of the multi-head self-attention [45]. Refinement attention mechanism is formulated as follows and the detailed structure is illustrated in Figure 4:

$$\text{score}(\tilde{h}_j^S, h_n^D) = v_j^\top \tanh(W_j^Q \tilde{h}_j^S + W_j^K h_n^D + \hat{b}_j^A), \tag{11}$$

$$\alpha_{j,n} = \frac{\exp(\text{score}(\tilde{h}_j^S, h_n^D))}{\sum_{n'=1}^N \exp(\text{score}(\tilde{h}_j^S, h_{n'}^D))}, \tag{12}$$

$$\tilde{h}_j^D = \sum_{n=1}^N \alpha_{j,n} W_j^V h_n^D, \tag{13}$$

where $1 \leq j \leq J$, $\tilde{h}_j^S$ is calculated with Equation (5), h_n^D is calculated with Equation (10), W matrices denote weight matrices, v_j is the weight vector, and $\hat{b}_j^A$ is the bias vector. $\tilde{h}_j^D$ is the intent representation in the j-th aspect. Reviewing Figure 1, we can obtain user intent representations $\tilde{h}_1^D$ and $\tilde{h}_2^D$ with the help of $\tilde{h}_1^S$ and $\tilde{h}_2^S$, respectively.

4.4. Preference and Intent Fusing

The significance of preferences/intents is uncertain. We propose a multi-aspect gated fusion mechanism to balance the influence of preferences and intents. The mechanism considers characteristics of preferences and intents in

16

each aspect and constructs the hybrid user representation $\tilde{h}_j^H$ accordingly:

$$\beta_j = \sigma(w_j^\top [\tilde{h}_j^S, \tilde{h}_j^D] + \hat{b}_j^F), \tag{14}$$

$$\tilde{h}_j^H = (1 - \beta_j)\tilde{h}_j^S + \beta_j \tilde{h}_j^D, \tag{15}$$

where $1 \le j \le J$, w_j is the weight vector, and $\hat{b}_j^F$ is the bias. Reviewing Figure 1, we can integrate $\tilde{h}_1^D$ and $\tilde{h}_1^S$, resulting in the generation of $\tilde{h}_1^H$. Similarly, by integrating $\tilde{h}_2^D$ and $\tilde{h}_2^S$, we can obtain $\tilde{h}_2^H$.

Transformer [45] proposes leveraging a concatenation and a projection to integrate parallel heads. Similarly, we concatenate hybrid user representations in different aspects (i.e., semantic subspaces) and utilize a projection again. In this way, we obtain the final user representation h^F as the following equation shows:

$$h^F = W^\rho[\tilde{h}_1^H, \tilde{h}_2^H, \cdots, \tilde{h}_J^H], \tag{16}$$

where the projection is a parameter matrix W^ρ. The dimensions of h^F and $\tilde{h}_\star^H$ are equal. Reviewing Figure 1, we can derive the final user representation h^F of User 1 based on $\tilde{h}_1^H$ and $\tilde{h}_2^H$.

Inspired by studies [48, 36], we predict next item and next category that a user is interested in with the following equations:

$$\hat{y}^I = \text{softmax}(W^{O,I} h^F), \tag{17}$$

$$\hat{y}^C = \text{softmax}(W^{O,C} h^F), \tag{18}$$

where $W^{O,I}$ and $W^{O,C}$ are the weight matrices. $\hat{y}^I$ and $\hat{y}^C$ are the predicted probabilities of items and categories, respectively.

4.5. Model Training

To augment the training data, we predict next item i_{m+1} and category c_{m+1} at every time step m. Thus, the total loss function is given by the following equation:

$$\mathcal{L}(\Theta) = -\sum_{u \in U} \sum_{m=1}^{M} \left(y_{m+1}^I \log(\hat{y}_{m+1}^I) + \gamma y_{m+1}^C \log(\hat{y}_{m+1}^C) \right) + \lambda \|\Theta\|^2, \tag{19}$$

17

where Θ denotes all trainable parameters, γ is the hyper-parameter which controls the effectiveness of category information, and λ is the hyper-parameter which controls the strength of the L2 regularization to prevent overfitting. Besides the L2 regularization, we adopt dropout [49] in the fusing component.

We optimize all models with Adam optimizer [50], which is an adaptive learning rate optimization algorithm.

5. Experiments

Next, we answer the following research questions:

- **RQ1:** How does MAINT perform compared with various advanced recommender models?

- **RQ2:** Are the support types of behaviors helpful for improving the performance of predicting the next target behavior?

- **RQ3:** How do different designs contribute to the model performance?

- **RQ4:** How do hyper-parameters affect the performance?

- **RQ5:** How is the interpretation ability of our model?

5.1. Experimental Settings

5.1.1. Datasets and Preprocessing

We utilize two real-world e-commerce datasets, i.e., Taobao [2] and Retailrocket [3], which contain multiple behavior types. Dataset statistics are shown in Table 4. Similar to previous studies [10, 27], we filter out users who have fewer than ten interactions and items that appear fewer than twenty times for Taobao. For Retailrocket, we filter out users and items with fewer than five and ten records, respectively. In addition, each user should have at least one

---

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=46
[3]https://www.kaggle.com/retailrocket/ecommerce-dataset

Table 4: Dataset statistics.

Dataset	Taobao		Retailrocket	
Users	10,000		1,407,580	
Items	2,876,947		417,053	
Categories	8,916		1,086	
Interactions	11,550,581	Clicks	2,664,312	Views
	242,556	Collects	69,332	Carts
	343,564	Carts	22,457	Buys
	120,205	Buys		
# Behavior Types	4		3	

buy record. Category information provided by Retailrocket is hierarchical (tree-like). In our experiments, we only use category information of the lowest level. The maximum length of sequences is twenty in our experiments.

5.1.2. Evaluation Protocols

Following the leave-one-out strategy applied widely on existing studies [42, 2], we utilize the most recently bought item of each user for testing, the second recently bought item for validation, and the rest bought items for training. For efficient evaluation, each positive instance is paired with 100 non-interactive items [12, 42]. A good recommender system should recommend new items that a user has not consumed before [2]. So for each user, we delete her training records with the test item.

To evaluate the performance of each model, we use two metrics: Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K). K is the number of recommended items. There are two primary reasons for using these two metrics. First, these metrics have been widely used by previous researchers [42, 2], and employing the same metrics enables a fair comparison of models. Second, these metrics focus on different perspectives: NDCG is a ranking-based measure that focuses on the position of the test item in the Top-K recommendation list, while HR is a recall-based measure that determines

whether the test item appears in the Top-K recommendation list. Together, they provide a comprehensive evaluation of a model from different perspectives.

5.1.3. Methods for Comparison

We compare our proposed method with three groups of recommender models:

Multi-behavioral Non-sequential Recommender Models.

- **BF** [14]. This method jointly factorizes matrices of different behavior types with sharing item-side embeddings.

- **MC-BPR** [15]. This method adopts sampling rules considering levels of different behavior types.

- **NMTR** [2]. This method considers the cascading relationship among different behavior types. It shares user-side and item-side embeddings and learns a separate interaction function for each behavior type.

Single-behavioral Sequential Recommender Models.

- **GRU4Rec** [25]. This method uses a GRU to model behaviors.

- **MCPRN** [27]. This method uses multiple RNNs to learn multiple purposes and proposes a purpose router to decide which RNN should be updated with an interacted item. Vanilla MCPRN does not consider category information. To ensure a fair comparison, we extend it by concatenating item embeddings and category embeddings as the input.

- **HGN**[38]. This method designs a feature gating and an instance gating to select features and items, respectively. We extend vanilla HGN to consider category information, too.

Multi-behavioral Sequential Recommender Models.

- **ATRank** [39]. This method groups behaviors by behavior type and chooses a self-attention network to model the influence among different behavior types.

- **BINN** [10]. This method exploits an LSTM-based model to learn from the preference behaviors and the session behaviors. We extend vanilla BINN to consider category information.

- **ASLI** [12]. This method learns item similarities based on the interacted item sequence via a transformer layer and obtains the user's intent based on her actions on a particular category via a convolution layer.

- **IARS** [40]. This method leverages multiple GRUs to learn diverse intentions. It designs three tasks to improve recommendation performance.

HGN [38] and ASLI [12] have outperformed some sequential models (such as FPMC [22], Fossil [23], Caser [29], NextItNet [30], SASRec [31]). We omit comparisons against these models.

5.1.4. Implementation Details

We implement MAINT with TensorFlow. The item embedding size is fixed to 64 for all models, which is suitable for models to learn a strong representation. The batch size of samples is 512. The initial learning rate is 0.01. λ is set to 10^{-5}. The number of implicit aspects is 3 (i.e., $J = 3$). γ is 1. The dropout rate is 0.2. For all models, we tune their hyper-parameters and report the best performance.

5.2. Performance Comparison (RQ1)

To answer **RQ1**, we repeat each experiment 5 times by changing the random seeds, and average results are reported in Table 5. The two-tailed unpaired t-test is performed to detect significant differences between MAINT and the best baseline. We have the following findings:

- In the first group, NMTR outperforms BF and MC-BPR due to its ability to model complex nonlinear interactions between users and items with deep learning.

Table 5: Comparisons on Taobao and Retailrocket. ⋆ means significantly better than the best baseline ($p < 0.05$).

| Method | Taobao | | | | | |
| | K=2 | | K=6 | | K=10 | |
	HR	NDCG	HR	NDCG	HR	NDCG
BF	0.1142	0.1015	0.2053	0.1399	0.2670	0.1592
MC-BPR	0.1201	0.1009	0.2421	0.1532	0.3264	0.1794
NMTR	0.1224	0.1073	0.2502	0.1613	0.3327	0.1871
GRU4Rec	0.0989	0.0850	0.2113	0.1328	0.2971	0.1600
MCPRN	0.2753	0.2564	0.3968	0.3114	0.4621	0.3328
HGN	0.2552	0.2381	0.3898	0.2958	0.4655	0.3210
ATRank	0.2588	0.2275	0.3753	0.2857	0.4459	0.3111
BINN	0.2871	0.2658	0.3945	0.3272	0.4599	0.3562
ASLI	0.2186	0.2019	0.3626	0.2754	0.4424	0.3027
IARS-S	*0.3020*	*0.2740*	*0.4095*	*0.3278*	*0.4857*	*0.3568*
MAINT	**0.3078⋆**	**0.2773⋆**	**0.4285⋆**	**0.3289⋆**	**0.5130⋆**	**0.3582⋆**

| Method | Retailrocket | | | | | |
| | K=2 | | K=6 | | K=10 | |
	HR	NDCG	HR	NDCG	HR	NDCG
BF	0.1039	0.1028	0.2174	0.1363	0.3104	0.1654
MC-BPR	0.1953	0.1710	0.3396	0.2337	0.4193	0.2587
NMTR	0.2680	0.2387	0.4166	0.3022	0.5041	0.3295
GRU4Rec	0.1030	0.0965	0.1827	0.1245	0.2323	0.1436
MCPRN	0.3071	0.2944	0.4969	0.3828	0.6106	0.4207
HGN	0.3827	0.3461	0.5574	0.4215	0.6305	0.4443
ATRank	0.3686	0.3573	0.5363	0.4246	0.6413	0.4525
BINN	0.3564	0.3418	0.5228	0.4151	0.6231	0.4370
ASLI	0.4165	0.3862	0.5527	0.4460	0.6423	0.4698
IARS-S	*0.4179*	*0.3875*	*0.5634*	*0.4482*	*0.6444*	*0.4735*
MAINT	**0.5187⋆**	**0.4802⋆**	**0.6525⋆**	**0.5450⋆**	**0.7175⋆**	**0.5693⋆**

- In the second group, MCPRN performs better than GRU4Rec. Since MCPRN divides items into several subsets according to user latent pur-

poses and models each subset of items with a variant GRU. It could capture subset-level dependencies besides item-level dependencies. Compared with NMTR, although it ignores the behavioral diversity, it captures the sequence-related patterns of the item- and subset-level well and performs better. HGN also performs better than GRU4Rec and NMTR. HGN incorporates a feature gating, an instance gating, and an aggregation layer to capture feature-level, instance-level, and group-level relations between past and future interaction items. It also utilizes an item-item product module to capture item-item relations. It has better performance than MCPRN on Retailrocket. The reason may be that gating networks are good at processing sparse data.

- IARS performs best among baselines since it could model multiple coexisting intentions with multiple GRUs and attention layers. Besides, auxiliary tasks are helpful for solving the main task. BINN could learn users' historical preferences and present motivations considering both multi-behavioral and sequence-related patterns. ASLI could learn users' intents from their actions on a particular category. ATRank and ASLI have better performance than BINN on Retailrocket. The reason may be that ATRank and ASLI have attention-based units which are better at handling sparse data.

- MAINT achieves significant improvement over all baselines on all datasets in both HR and NDCG. Compared with MCPRN and ASLI, MAINT mainly can capture more aspects of user preferences/intents. Compared with HGN and BINN, MAINT excels in its ability to adaptively fuse preferences and intents while learning them from multiple aspects. Compared with ATRank, MAINT effectively models sequence-related patterns existing in the multi-typed behavioral sequence. Compared with IARS, MAINT can learn multi-aspect preferences and filter out noises existing in the multi-typed behavioral sequence.

| (a) Taobao | (b) Retailrocket |

Figure 5: Impact study of different support types.

5.3. Impact of support types of behaviors (RQ2)

To answer **RQ2**, we design variants of our model. These variants are generated as follows: "-" behavior type means removing behaviors of this type. From the results in Figure 5, we can see MAINT with all support types of behaviors achieves the best performance, which means support types of behaviors are helpful for improving prediction performance. Among support types, *click* and *view* are the most important behavior types to help MAINT improve performance. That is probably because behaviors of *click* and *view* account for a significant proportion.

5.4. Ablation Study (RQ3)

To prove the effectiveness of the key designs in MAINT and answer **RQ3**, we consider different model variants from four perspectives:

Table 6: Ablation study of key designs in MAINT.

Model	Taobao		Retailrocket	
	HR@10	NDCG@10	HR@10	NDCG@10
MAINT-MP	0.4977	0.3471	0.7144	0.5560
MAINT-BLSTM	0.4977	0.3465	0.7066	0.5600
MAINT-RAtt	0.4876	0.3482	0.7109	0.5550
MAINT-MGFus	0.4925	0.3470	0.6938	0.5302
MAINT	0.5130	0.3582	0.7175	0.5693

- Effect of the Multi-aspect Projection Mechanism. MAINT-MP. We consider one variant of MAINT without the projection mechanism. In other words, we utilize one latent vector to represent a user's preferences/intents (i.e., $J = 1$).

- Effect of Behavior-enhanced LSTM. MAINT-BLSTM. We do an ablation study to test the effectiveness of Behavior-enhanced LSTM by comparing the performance of the model with vanilla LSTM.

- Effect of Refinement Attention Mechanism. MAINT-RAtt. We replace Refinement Attention Mechanism with vanilla attention [51].

- Effect of Multi-aspect Gated Fusion Mechanism. MAINT-MGFus. We replace Multi-aspect Gated Fusion Mechanism with concatenation.

Table 6 reports the results. We can find the full version of MAINT achieves the best performance in all cases. This proves that the key designs work. The reasons why our designs work have been introduced in the previous section.

5.5. Impact of Hyper-parameters (RQ4)

For answering **RQ4**, we select four important hyper-parameters to study:

- Number of implicit aspects J. We adjust the number of aspects and the results are shown in Figure 6. As the number of aspects increases,

(a) Taobao. (b) Retailrocket.

Figure 6: Study of # aspects.

(a) Taobao.

(b) Retailrocket.

Figure 7: Study of embedding dimensionality.

(a) Taobao.

(b) Retailrocket.

Figure 8: Study of γ.

the performance grows in NDCG. When the number of aspects is 4, the performance drops. We think the overfitting problem may occur.

- Embedding dimensionality. We increase embedding dimensionality from 8 to 64, the model achieves better performance as Figure 7 shows. Because of the overfitting phenomenon, the performance degrades a little in NDCG with the further increase of embedding dimensionality.

- Coefficient γ. Figure 8 shows the results. Next-category recommendation task plays an important role in enhancing the performance of next-item recommendations. In general, the model performs better with larger values of γ. The reason may be that next-category recommendation task helps the model learn user preferences/intents from the category-aspect.

(a) Taobao. (b) Retailrocket.

Figure 9: Study of the maximum length of sequences.

- Maximum length of sequences N. We conduct experiments by varying the sequence length between 5 and 40, and the results are presented in Figure 9. We observed that as the length increases, the performance of MAINT initially improves but then declines. When the length is 10, MAINT achieves the best performance in NDCG on Taobao. When the length is 30, MAINT achieves the best performance in NDCG on Retailrocket. When the length is 20, MAINT achieves the best performance in HR on both datasets. It indicates that a longer sequence may include too many irrelevant items for predicting future items. In other words, the preference/intent drift problem is serious in a longer sequence. A larger length might lead to heavy computational costs at the same time. Meanwhile, the average lengths of processed sequences are about 19 and 17 on Taobao and Retailrocket, respectively. So the maximum length of sequences is 20 in our experiments.

5.6. Case Study of model's Interpretability (RQ5)

In response to **RQ5**, we conduct a case study by randomly selecting a user from the Retailrocket dataset. The attention scores α and gate scores β are visualized in Figure 10. In this case, our model successfully recommends both the item and the category. Based on 20 behaviors of the user, our model initially recommends item i23930, which the user subsequently purchases. Our observations are as follows:

27

Figure 10: A real example from a sampled user on Retailrocket.

- MAINT focuses on different behaviors in different aspects since the distributions vary a lot in 3 aspects. In the 3rd aspect, the attention scores of the 17th, 18th, 19th, and 20th behaviors are high, and the categories of these items align with the predicted item. This suggests that the 3rd aspect may represent the category-aspect. However, due to insufficient information, it is not possible to infer the specific meanings represented by the other aspects.

- MAINT is capable of balancing preferences and intents, as indicated by the variation in gate scores across 3 aspects. In the 3rd aspect (i.e., the category-aspect), the gate score is high. This could be because the user made few purchases a long time ago but clicked many items recently, indicating that her intents may be more significant.

The experiment demonstrates the effectiveness of both the attention mechanism and the gated fusion mechanism in interpreting recommendation results.

6. Discussion

In this section, we discuss the extensibility and efficiency of our model.

6.1. Model Extensibility

We would like to emphasize that our proposed model serves as a general recommender model, given its ability to model multi-typed behaviors. To show

28

(a) (b)

Figure 11: Comparisons on Movielens-1M.

the extensibility of our model, we utilize another dataset: Movielens-1M [4].

Movielens-1M is a popular dataset [52, 31]. It consists of 1,000,209 ratings for approximately 3,900 movies, provided by 6,040 users. The ratings are categorized into five levels (1, 2, 3, 4, 5), which can be viewed as five types of behavior. Our goal with this dataset is to predict the next movie that a user will rate 5 stars. For movies with multiple genre (i.e., category) tags, we only use the most frequently occurring one in the dataset. We also limit the sequence length to a maximum of 20.

We found that the model achieved the best results under the existing hyperparameter settings. To provide a comparative analysis, we selected the most competitive baseline (i.e., IARS-S). The results are depicted in Figure 11, clearly demonstrating that our model continues to perform exceptionally well.

6.2. Model Efficiency

We then analyze the time complexity of our model. The time complexity of the shared embedding layer is $O(N)$. The time complexity of the multi-aspect projection mechanism is $O(Jd^2)$, where d is the embedding dimensionality. The time complexity of two LSTMs is $O(Md^2 + Nd^2)$, where $M < N$. The time complexity of the multi-aspect refinement attention mechanism is $O(JNd^2)$. The time complexity of the multi-aspect gated fusion mechanism is $O(Jd+Jd^2)$.

---

[4]https://grouplens.org/datasets/movielens/1m/

(a) Offline Training.

(b) Online Recommending.

Figure 12: Time Comparison.

Therefore, the total time complexity of our model is $O(N + Jd^2 + Md^2 + Nd^2 + JNd^2 + Jd + Jd^2)$, i.e., $O(JNd^2)$. The total time complexity of IARS-S is $O(\Lambda Nd^2)$, where Λ is the number of GRUs. Figure 12 (a) compares the training time per batch of the best models (IARS-S and MAINT) in seconds. Figure 12 (b) compares the performance of these models in an online recommendation scenario, where the number of requests varies. We can see that the efficiency of MAINT is slightly worse than IARS-S. But the recommendation performance of MAINT is markedly superior. This limitation can be mitigated by utilizing machines with enhanced computational capabilities. On balance, the benefits of our model significantly outweigh its limitations.

We also analyze the space complexity of our model. The space complexity of the shared embedding layer is $O(\check{d}d)$, where $\check{d}$ is the sum of the dimensions of one-hot vectors. The space complexity of the multi-aspect projection mechanism is $O(Jd^2)$. The space complexity of two LSTMs is $O(d^2)$. The space complexity of the multi-aspect refinement attention mechanism is $O(Jd^2)$. The space complexity of the multi-aspect gated fusion mechanism is $O(Jd + Jd^2)$. Therefore, the space complexity of our model is $O(\check{d}d + Jd^2)$. The total space complexity of IARS-S is $O(\check{d}d + \Lambda d^2)$.

7. Conclusion

In this paper, we propose a novel model called MAINT for multi-behavioral sequential recommendation. The experiments demonstrate that our models consistently outperform state-of-the-art recommender models. In summary, the key contributions are as follows: we propose an approach to extract multi-aspect preferences from target behaviors; we design network structures to extract multi-aspect intents from multi-typed behaviors; and we develop a mechanism to adaptively fuse multi-aspect preferences and intents. Pre-training has proven effective in various fields. Therefore, we aim to design an item embedding pre-training method (based on graph neural networks) considering multi-typed behaviors in the future.

Acknowledgements

References

[1] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Comput. Surv. 51 (4) (2018) 66:1–66:36.

[2] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T.-S. Chua, L. Yao, Y. Song, D. Jin, Learning to recommend with multiple cascading behaviors, IEEE Trans. Knowl. Data Eng. 33 (6) (2021) 2588–2601.

[3] H. Liu, Y. Zhu, T. Zang, Y. Xu, J. Yu, F. Tang, Jointly modeling heterogeneous student behaviors and interactions among multiple prediction tasks, ACM Trans. Knowl. Discov. Data 16 (1) (2022) 16:1–16:24.

[4] H. Liu, Y. Zhu, C. Wang, J. Ding, J. Yu, F. Tang, Incorporating heterogeneous user behaviors and social influences for predictive analysis, IEEE Trans. Big Data 9 (2) (2023) 716–732.

[5] C. Wang, W. Ma, M. Zhang, C. Chen, Y. Liu, S. Ma, Toward dynamic user intention: Temporal evolutionary effects of item relations in sequential recommendation, ACM Trans. Inf. Syst. 39 (2) (2021) 16:1–16:33.

[6] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: UAI, 2009, pp. 452–461.

[8] M. Zhou, Z. Ding, J. Tang, D. Yin, Micro behaviors: A new perspective in e-commerce recommender systems, in: ACM WSDM, 2018, pp. 727–735.

[9] Q. Liu, S. Wu, L. Wang, Multi-behavioral sequential prediction with recurrent log-bilinear model, IEEE Trans. Knowl. Data Eng. 29 (6) (2017) 1254–1267.

[10] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, E. Chen, Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors, in: ACM SIGKDD, 2018, pp. 1734–1743.

[11] L. Guo, L. Hua, R. Jia, B. Zhao, X. Wang, B. Cui, Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior, in: ACM SIGKDD, 2019, pp. 1984–1992.

[12] M. M. Tanjim, C. Su, E. Benjamin, D. Hu, L. Hong, J. McAuley, Attentive sequential models of latent intent for next item recommendation, in: ACM WWW, 2020, pp. 2528–2534.

[13] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[14] Z. Zhao, Z. Cheng, L. Hong, E. H. Chi, Improving user topic interest profiles by behavior factorization, in: ACM WWW, 2015, pp. 1406–1416.

[15] B. Loni, R. Pagano, M. Larson, A. Hanjalic, Bayesian personalized ranking with multi-channel user feedback, in: ACM RecSys, 2016, pp. 361–364.

[16] B. Jin, C. Gao, X. He, D. Jin, Y. Li, Multi-behavior recommendation with graph convolutional networks, in: ACM SIGIR, 2020, pp. 659–668.

[17] L. Xia, C. Huang, Y. Xu, P. Dai, M. Lu, L. Bo, Multi-behavior enhanced recommendation with cross-interaction collaborative relation modeling, in: IEEE ICDE, 2021, pp. 1931–1936.

[18] C. Chen, W. Ma, M. Zhang, Z. Wang, X. He, C. Wang, Y. Liu, S. Ma, Graph heterogeneous multi-relational recommendation, in: AAAI, 2021, pp. 3958–3966.

[19] W. Zhang, J. Mao, Y. Cao, C. Xu, Multiplex graph neural networks for multi-behavior recommendation, in: ACM CIKM, 2020, pp. 2313–2316.

[20] C. Park, D. Kim, J. Han, H. Yu, Unsupervised attributed multiplex network embedding, in: AAAI, 2020, pp. 5371–5378.

[21] P. Yu, C. Fu, Y. Yu, C. Huang, Z. Zhao, J. Dong, Multiplex heterogeneous graph convolutional network, in: ACM SIGKDD, 2022, pp. 2377–2387.

[22] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: ACM WWW, 2010, pp. 811–820.

[23] R. He, J. J. McAuley, Fusing similarity models with markov chains for sparse sequential recommendation, in: IEEE ICDM, 2016, pp. 191–200.

[24] S. Hosseini, H. Yin, X. Zhou, S. W. Sadiq, M. R. Kangavari, N. Cheung, Leveraging multi-aspect time-related influence in location recommendation, WWWJ 22 (3) (2019) 1001–1028.

[25] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, in: ICLR, 2016.

[26] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, D. Cai, What to do next: Modeling user behaviors by time-lstm, in: IJCAI, 2017, pp. 3602–3608.

[27] S. Wang, L. Hu, Y. Wang, Q. Z. Sheng, M. A. Orgun, L. Cao, Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks, in: IJCAI, 2019, pp. 3771–3777.

[28] P. Zhao, A. Luo, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, X. Zhou, Where to go next: A spatio-temporal gated network for next poi recommendation, IEEE Trans. Knowl. Data Eng. 34 (5) (2022) 2512–2524.

[29] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: ACM WSDM, 2018, pp. 565–573.

[30] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, X. He, A simple convolutional generative network for next item recommendation, in: ACM WSDM, 2019, pp. 582–590.

[31] W.-C. Kang, J. McAuley, Self-attentive sequential recommendation, in: IEEE ICDM, 2018, pp. 197–206.

[32] T. Zhang, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, X. Zhou, Feature-level deeper self-attention network for sequential recommendation, in: IJCAI, 2019, pp. 4320–4326.

[33] Y. Luo, Q. Liu, Z. Liu, Stan: Spatio-temporal attention network for next location recommendation, in: ACM WWW, 2021, pp. 2177–2185.

[34] M. Saaki, S. Hosseini, S. Rahmani, M. R. Kangavari, W. Hua, X. Zhou, Value-wise convnet for transformer models: An infinite time-aware recommender system, IEEE Trans. Knowl. Data Eng. (2022) 1–12.

[35] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: AAAI, 2019, pp. 346–353.

[36] Y. Li, T. Chen, Y. Luo, H. Yin, Z. Huang, Discovering collaborative signals for next poi recommendation with iterative seq2graph augmentation, in: IJCAI, 2021, pp. 1491–1497.

[37] X. Rao, L. Chen, Y. Liu, S. Shang, B. Yao, P. Han, Graph-flashback network for next location recommendation, in: ACM SIGKDD, 2022, pp. 1463–1471.

[38] C. Ma, P. Kang, X. Liu, Hierarchical gating networks for sequential recommendation, in: ACM SIGKDD, 2019, pp. 825–833.

[39] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, J. Gao, Atrank: An attention-based user behavior modeling framework for recommendation, in: AAAI, 2018, pp. 4564–4571.

[40] Y. Xu, Y. Zhu, J. Yu, Modeling multiple coexisting category-level intentions for next item recommendation, ACM Trans. Inf. Syst. 39 (3) (2021) 23:1–23:24.

[41] Y. Wu, R. Xie, Y. Zhu, X. Ao, X. Chen, X. Zhang, F. Zhuang, L. Lin, Q. He, Multi-view multi-behavior contrastive learning in recommendation, in: DASFAA Part II, 2022, pp. 166–182.

[42] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: ACM WWW, 2017, pp. 173–182.

[43] H. Fang, D. Zhang, Y. Shu, G. Guo, Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations, ACM Trans. Inf. Syst. 39 (1) (2020) 10:1–10:42.

[44] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014) 1–9.

[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NeurIPS, 2017, pp. 5998–6008.

[46] R. Qiu, Z. Huang, T. Chen, H. Yin, Exploiting positional information for session-based recommendation, ACM Trans. Inf. Syst. 40 (2) (2022) 35:1–35:24.

[47] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: ECCV, Part I, 2014, pp. 818–833.

[48] L. Chen, Y. Liu, X. He, L. Gao, Z. Zheng, Matching user with item set: Collaborative bundle recommendation with deep attention network, in: IJCAI, 2019, pp. 2095–2101.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[50] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.

[51] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: HLT-NAACL, 2016, pp. 1480–1489.

[52] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2016) 19:1–19:19.

Appendices

Details on the Conversion Rate. The conversion rate of some behavior type is computed as:

$$Conversion\ Rate = \frac{\#\ behaviors\ of\ this\ type\ on\ each\ item\ before\ purchase}{\#\ behaviors\ of\ this\ type}.$$

Training Procedure of MAINT. The overall training procedure of MAINT is illustrated in Algorithm 1. Line 6-9 correspond to the multi-aspect preference modeling stage. Line 10-14 correspond to the multi-aspect intent modeling stage. Line 15-19 correspond to the preference and intent fusing stage.

Algorithm 1 Learning Algorithm for MAINT
___

Input:

    multi-typed behavioral sequence of all users $\{X^u = \{x_1, x_2, \cdots, x_N\}\}, u \in U$; item embedding size; number of implicit aspects J; category prediction loss coefficient γ; L2 regularization coefficient λ; dropout rate

Output:

    MAINT with learned parameters Θ

1: Initialize all trainable parameters Θ;

2: **while** stopping criteria is not met **do**

3:     Draw a mini-batch from $\{X^u\}$;

4:     **for** each u **do**

5:         $p_n, q_n, r_n, s_n = \text{Embedding}(i_n, c_n, b_n, \Delta t_n), \quad 1 \leq n \leq N$ (see Eq. (1), (2));

6:         $h_m^S = \text{LSTM}([p_m, q_m], h_{m-1}^S)$, the type of $\bar{x}_m$ is target behavior (see Eq. (3));

7:         **for** each $j \in [1, J]$ **do**

8:             $\tilde{h}_j^S = W_j^P h_M^S$ (see Eq. (5));

9:         **end for**

10:        $h_n^D = \text{Behavior-enhanced\_LSTM}(p_n, q_n, r_n, s_n), 1 \leq n \leq N$ (see Eq. (6),(7),(8),(9),(10));

11:        **for** each $j \in [1, J]$ **do**

12:           $\alpha_{j,n} = \text{Refinement\_Attention\_Mechanism}(\tilde{h}_j^S, h_n^D), \quad 1 \leq n \leq N$ (see Eq. (12));

13:           $\tilde{h}_j^D = \sum_{n=1}^N \alpha_{j,n} W_j^V h_n^D$ (see Eq. (13));

14:        **end for**

15:        **for** each $j \in [1, J]$ **do**

16:           $\beta_j = \sigma(w_j^\top [\tilde{h}_j^S, \tilde{h}_j^D] + \hat{b}_j^F)$ (see Eq. (14));

17:           $\tilde{h}_j^H = (1 - \beta_j)\tilde{h}_j^S + \beta_j \tilde{h}_j^D$ (see Eq. (15));

18:        **end for**

19:        $h^F = W^\rho [\tilde{h}_1^H, \tilde{h}_2^H, \cdots, \tilde{h}_J^H]$ (see Eq. (16));

20:        $\hat{y}^I = \text{softmax}(W^{O,I} h^F), \quad \hat{y}^C = \text{softmax}(W^{O,C} h^F)$ (see Eq. (17), (18));

21:     **end for**

22:     Compute loss according to $\mathcal{L}(\Theta) = -\sum_{u \in U_{batch}} \sum_{m=1}^M \left(y_{m+1}^I \log(\hat{y}_{m+1}^I) + \gamma y_{m+1}^C \log(\hat{y}_{m+1}^C) \right) + \lambda \|\Theta\|^2$;

23:     Update Θ according to $\mathcal{L}$ and optimizer;

24: **end while**

25: **return** Θ
___