Journal of Big Data

RESEARCH Open Access

RTiSR: a review-driven time interval-aware sequential recommendation method

Xiaoyu Shi[1,3], Quanliang Liu[1,3], Yanan Bai[2] and Mingsheng Shang[1,3*]

*Correspondence:
msshang@cigit.ac.cn

[1] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
[2] School of Artificial Intelligence, Chongqing University of Technology, Chongqing, China
[3] Chongqing School, University of Chinese Academy of Sciences, Chongqing, China

Abstract

The emerging topic of sequential recommender (SR) has attracted increasing attention in recent years, which focuses on understanding and learning the sequential dependencies of user behaviors hidden in the user-item interactions. Previous methods focus on capturing the point-wise sequential dependencies with considering the time evenly spaced. However, in the real world, the time and semantic irregularities are hidden in the user's successive actions. Meanwhile, with the tremendous increase of users and items, the hardness of modeling user interests from spare explicit feedback. To this end, we seek to explore the influence of item-aspect reviews sequence with varied time intervals on sequential modeling. We present RTiSR, a review-driven time interval-aware sequential recommendation framework, to predict the user's next purchase item by jointly modeling the sequence dependencies from aspect-aware reviews. The main idea is twofold: (1) explicitly learning user and item representation from reviews by assigning different weights, and (2) leveraging a hybrid neural network to capture the collective sequence patterns with a flexible order from aspect-aware review sequences. We conduct extensive experiments on industrial datasets to evaluate the effectiveness of RTiSR. Experimental results demonstrate the superior performance of RTiSR in different evaluation metrics, compared to the state-of-the-art competitors.

Keywords: Recommender system, Sequential recommendation, Review-driven, Deep learning, Time-aware model

Introduction

In the big data era, recommender systems (RSs) play an important role in helping users find potential items of interest via sifting from massive choices [1, 2]. RSs have been widely applied in online shopping websites, content-share platforms, social networking and etc. The traditional RSs with the representation of collaborative filter (CF) [3, 4] and content-based [5] methods that focus on modeling the user-item interactions in a static view, thus these methods can only learn the general preferences of users. Compared to the traditional RSs, the emerging topic of sequential recommender systems (SRSs) aims to predict the next item or next few items successively based on users' sequential interaction behaviors, which can be easy to capture the sequential dependencies for more accurate recommendations. As a result, the sequential recommendation has received increasing attention in recent years [6, 7].

The key challenge of SRSs is to dynamically estimate the user's current preference by adaptive modeling the user's sequential patterns in historical interactions. Following this line, various methods have been proposed to learn the sequential patterns in user historical interactions, such as Markov Chains (MC)-based models [8, 9], and the recurrent neural networks (RNN)-based models [10]. For MC-based methods, they utilize a K-gram Markov chain to model the interactions between user and item in a sequence, for predicting the next item the user will purchase. FM-based methods usually utilize matrix factorization or tensor factorization to factorize the observed user-item interactions into latent factors of users and items for recommendation. Regarding of DNN-based methods, CNN-based methods utilize the convolutional filter and sliding windows strategies to capture the short-term contexts for prediction. RNN-based methods try to estimate the next possible interaction via modeling the sequential dependencies over the given interactions, which usually adopt the gated-recurrent (GRU) or long short-term memory (LSTM) units to learn the sequential dependencies hidden in the user-item interactions.

Although the previous solutions have achieved satisfactory results, they commonly suffer from the following defects:

- **The hardness of modeling the user preferences and item features from spare explicit feedback**. In most past studies, ratings are used as the only criterion of feedback information to measure the degree of user preference for the specified item. However, the ratings only reflect the overall satisfaction over an item without further details. As shown in Fig. 1, on the website Amazon both users buy the album "California Girls" with positive reviews, while user1 also buys the album "Pet Sounds" but give a negative review. Without consideration of the reviews, existing rating-based methods would recommend "Pet Sounds" to the user2 as it was rated by user1. However, it turns out improper recommendation. Thus, the

Fig. 1 A toy user-item graph of two Amazon users and albums. The improper recommendation is happened when only considering user connectivity but ignoring the reviews

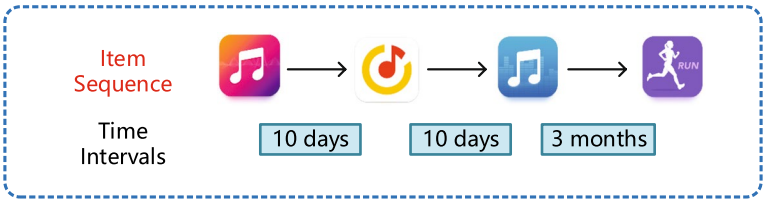


Fig. 2 A user's successive actions on Amazon's Software store, and time interval information is also given

improper recommendation is happened when only considering user connectivity but ignoring the reviews.

- **The semantic and item irregularities are hidden in the user's successive actions**. For the majority of users, the sequential dependencies of their interaction behaviors are not strictly ordered. Due to the uncertainty of user shopping behaviors, the main purpose behind the user behavior sequence is not clean. Thus, in the real world, some user behavior sequences are not strictly ordered, i.e., not all adjacent interactions are sequentially dependent in a sequence. For instance, as shown in Fig. 2, given the historical interaction sequence of a user as: S = {Music_ Player 1, Music_Player 2, Music_Player 3, Sports Tracker}. It seems that the first three items in S indicate that the user has a higher probability of buying a music player next, than buying sports tracker software. However, it is not a valid recommendation, since the user chooses the sports tracker app after three months. Hence, this kind of temporal distance deserves specific handling.

Hence, the research question arises:

R.Q. How to capture the collective sequential dependencies with a flexible orders become a key challenge in sequential recommendation domain?

To tackle the above defects, we propose an Review-driven Time interval aware Sequnetial Recommendation framework, named RTiSR, to predict user's next purchase item via modeling dynamical preferences of user and item extracted from aspect-aware reviews:

- First, compared to the ratings, review texts are not only much more expressive than ratings, but also provide a strong tool to explain the underlying dimensions behind users' decisions. On that basis, we employ the reviews to build the sequential recommendation model with more accurate user and item representations. Specifically, user-provided reviews can be viewed from the aspects of user and item. From the user aspect, a user's review set reflects the experience of buying diverse items. From the item aspect, an item's review set includes the reviews written to an item, and usually exhibits the various features of the specific item. Therefore, learning the user and item representations from reviews has a strengthening effect on sequential recommendation. In Fig. 3, we can observe that user u has a diverse purchase behavior from the user-aspect reviews, and reflects the user's preference more accurately. And the item-aspect reviews exhibit the various features of the specific item v from different users. In this case, the match-

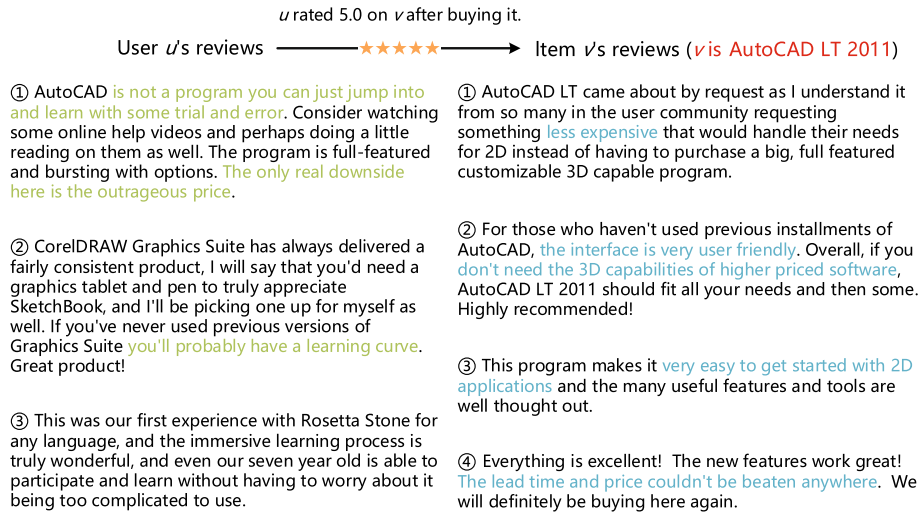


Fig. 3 A toy user-item graph of two Amazon users and albums. The improper recommendation is happened when only considering user connectivity but ignoring the reviews

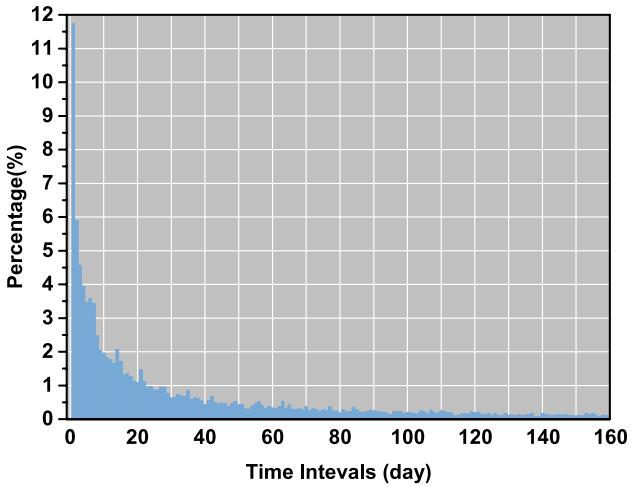


Fig. 4 Distribution of time intervals between user's adjacent reviews in the Amazon dataset

ing aspect reviews significantly help recommend models to predict the user's next purchase item, and u indeed marked a 5.0 score on v after purchasing it.

- Second, we also notice that there is no strict order and time between user sequential behaviors due to the uncertainty of user behaviors. The existing time-aware sequential recommenders [11, 12] always assume that the items in sequence can be considered evenly spaced and semantically consistent. However, in the practical recommendation scenarios, the user's behavior sequence is complex. As shown in Fig. 4, the time intervals between two adjacent reviews can be various, with the max interval of two adjacent reviews being about 160 days past. Intuitively, the two actions within a short time interval tend to share a closer relationship

than two actions within a long time interval. Thus *this kind of temporal distance deserves special handling*. To do it, we leverage a couple of convolutional fitters with varying sizes to effectively learn the user and item latent factors with flexible order.

Specifically, RTiSR consists of two homogeneous hybrid neural networks named U-Net and I-Net, which combine a recurrent neural network and a convolutional neural network. The U-Net focuses on exploiting user representation based on the reviews written by the user, while I-Net captures item multiple features from the reviews written for items. Then, a user-item interaction model (e.g., Factorization Machine) is applied to model the complex user-item interactions and output the predictions. The contributions of this paper are threefold:

- We propose RTiSR, a novel review-driven time interval-aware framework that exploits reviews with time interval information for a sequential recommendation, which captures sequence dependencies from the aspect-aware review sequence respectively.
- We introduce the flexible sequential pattern learning layer to learn the collective sequential dependencies with flexible order. We regard the embedded sequential reviews with explicit time interval information as an image, then employ multi-size convolution fitters to capture the collective sequential dependencies with flexible order, rather than the point-wise way.
- We conduct extensive experiments deployed on five real-world datasets. The experimental results demonstrate that the RTiSR achieves competitive and superior HR/NDCG results, compared to SOTA methods.

The remainder of this paper is organized as follows. Section summarizes the related work. Section gives the problem statement formally and describes the proposed RTiSR in detail. Section presents the experiment details. Section concludes our paper.

Related work

Sequential recommender

User's shopping behaviors usually successively in a sequence, rather than in an isolated manner. Thus, the key to accurate recommendation is properly capturing user dynamic preferences based on their historical interactions. Many approaches have been proposed to model users' interactions in a sequential manner for prediction or recommendation. The first line of research is that introducing a K-gram Markov chain to model the interactions between user and item in a sequence, for predicting the next item the user will purchase. For instance, Garcin et al. [13] adopt the first-order Markov Chains to capture the sequential pattern from user's browsing behaviors. Redle et al. [9] proposed FPMC to predict user's next-basket behaviors via factorizing the Markov Chains of user behaviors

with tensor factorization method. He et al. [14] proposed Fossil that integrates a similarity model into the high-order MC for the next item recommendation. Recently, recent neural network (RNN) played a dominant role in sequential recommendation, due to their success in sequence modeling on the natural language process (NLP) domain. They try to estimate the next possible interaction via modeling the sequential dependencies over the given interactions, which usually adopt the gated-recurrent (GRU) [15] or long short-term memory (LSTM) [10] units to learn the sequential dependencies hidden in the user-item interactions. For example, GRU4Rec [15] first utilize the RNN network to learn the sequential patterns. Quadrana et al. [16] propose a hierarchical RNN network to capture the cross-session dependencies in user behavior sequence. However, the high-order Markov chain model involves limited historical information, while the RNN-based methods are built on the strict order assumption.

To cope with the shortcomings of MC-based and RNN-based models, several attention-based RNN methods have been proposed to handle the noise or irrelevant interactions in user behavior sequence [7, 17, 18]. To model the union-level and point-level sequential patterns, Tang et al. proposed *Caser*, a convolutional sequence embedding recommendation model, which utilizes the convolutional filter with horizontal and vertical sliding windows strategies to capture the short-term contexts for prediction [19]. Li et al. proposed TiSASRec [12] that leverages a self-attention model for sequential recommendation, which can adaptive assign varying weights to different items by considering the absolute position and time interval in a sequence. Although the above mentioned solutions have achieved satisfactory results, the rich semantic information hidden in user-provided reviews are seriously overlooked in sequential recommendation. However, the above mentioned methods with two aspects: (1) they focus on capturing the point-wise dependencies only while ignoring the collective dependencies; (2) Due to the strong assumption that any adjacent interactions must be dependent, the RNN-based methods perform well on dense datasets, but show poor performance on sparse datasets.

Similar to our work, Li et al. proposed a review-driven neural model that captures the union-level and individual-level sequential dependencies by exploiting reviews [20]. Liu et al. [21] proposed an end-to-end neural network model to capture users' long-term and short-term preferences. However, they focus on capturing the sequence patterns on the user-aspect review sequence and ignore the importance of temporal dynamical of the item-aspect review sequence.

Review-based recommender

User reviews have been introduced as one of the important approaches to improve user and item representations in the recommendation domain, and have received lots of attention in RS research [20, 22, 23]. In earlier, some solutions focus on building a topic model to extract latent topics from reviews [24–26]. For instance, McAuley and Leskovec [24] adopt the Latent Dirichlet Allocation (LDA) model to discover users' and items' latent spaces from reviews. Tan et al. [26] tried to capture user preference with ratings and reviews, by mapping user preferences and item features into a latent topic space. However, these above methods consider the review in a bag-of-words representation and ignore the semantic information hidden in the reviews.

Table 1 Notations

Symbols	Definitions
$U, \mathcal{V}$	The user set and item set
$u \in U, v \in \mathcal{V}$	A user and an item
$\mathcal{S}^u, \mathcal{S}^v$	The behavior sequences of user u and item v
$\mathcal{D}^u, \mathcal{D}^v$	Review sequence of user u and item v corresponding to $\mathcal{S}^u, \mathcal{S}^v$
T^u, T^v	Timestamp sequence of user u and item v corresponding to $\mathcal{S}^u, \mathcal{S}^v$
D_v^u	User u writes a review on item v
$y_v^u, \hat{y}_v^u$	The true and estimated rating
$\mathbf{o}^u, \mathbf{o}^v$	User and item representation with size of d_o
m	The number of words in a review text
n_c	The number of convolution filters
d_w	The dimension of each word vector
d_l	The dimension of hidden state in LSTM
d_o	The dimension of user/item representation
$\oplus$	The concatenation operator
$\odot$	The element-wise product operator

Recently, several methods focus on employing deep learning techniques to capture user preferences from reviews for a recommendation. Zheng et al. [27] proposed DeepCoNN, a neural network-based framework that learns the user preferences and item features from reviews jointly. Based on DeepCoNN, Catherine et al. [28] proposed TransNets that insert an additional latent layer to represent the user-item pair. Tay et al. [29] propose MPCN, a multi-pointer co-attention neural network by exploiting reviews. Besides, Wu et al. [30] proposed CARL that derives two separate learning components from exploiting review data and interaction data. By leveraging the user-provided reviews, these above-mentioned methods achieved competitive performance with a more accurate user and item representation. However, the above-mentioned methods focus on the rating prediction task and do not consider enhancing sequential recommendation with user-provided reviews.

Methodology

In this section, we first give a formal problem statement of a review-driven sequential recommendation task and then depict the details of the proposed RTiSR.

Problem statement

For the convenience of expression, we first introduce some notations used in this paper, as shown in Table 1. Then we formalize the relevant definitions and problems as follows:

Definition 1 (*Interaction record*) Given user set $\mathbf{U} = \{u_1, u_2, \ldots, u_{|\mathbf{U}|}\}$, item set $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, an interaction record $s = (u, v, D_v^u, y_v^u, t_v^u)$ is a five-elements tuple, which means user u rated on item v with given a review D_v^u, and rating y_v^u at time t_v^u.

Definition 2 (*Interaction sequence*) Let S^u be the chronologically ordered behavior sequence of a user u, then it can formally define as $S^u = \{s_1^u, s_2^u, \ldots, s_{|S^u|}^u\}$, where

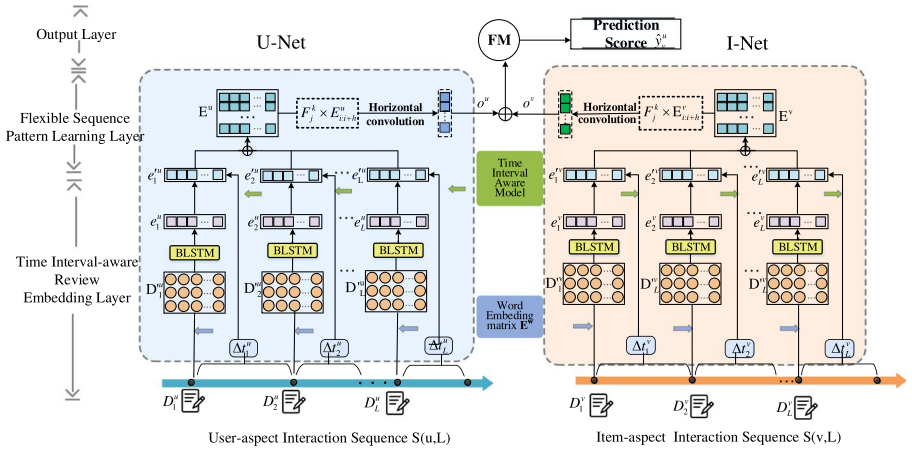


Fig. 5 Illustration of our proposed model

s_t^u denotes the user u happens a interaction record at time t. Similarly, the behavior sequence happened on an item v can also be defined as $S^v = \{s_1^v, s_2^v, \cdots, s_{|S^v|}^v\}$.

Definition 3 (*Review-driven sequential recommendation*) Given the recently purchased L items of user u and the corresponding review documents $\mathcal{D}^{u,L}$, the goal is to generate a recommendation list with top ranked candidate items via computing the likelihood she will purchase in the next.

The architecture of RTiSR

In this work, we seed to exploit user-provided reviews to enhance the performance of sequential recommendations with implicit feedback. Figure 5 depicts the network architecture of RTiSR. The objective of RTiSR is to derive the user and item collective sequential patterns with a flexible order from the aspect-aware review sequences. To do it, RTiSR consists of two homogeneous neural networks named U-Net and I-Net. RTiSR includes three layers: the time interval-aware review embedding layer, the flexible sequence pattern learning layer, and the prediction layer. Specifically, the review embedding layer organizes the user-provided review set to the aspect-aware review sequences, then encodes each review into an embedding vector by utilizing a Bi-directional Long Short-Term Memory (BiLSTM) network. After that, the time interval model is proposed to assign dynamical weights to different reviews in a sequence. Then the flexible sequence pattern learning layer assembles the learned time interval-aware review embedded as an "image", and adopts a set of convolution kernels with varied sizes to learn the flexible sequential patterns. In the prediction layer, a factorization machine (FM) model is adopted to model the complex user-item interaction and obtain recommendations.s adopted to model the complex user-item interaction and obtain recommendations.

The time interval-aware review embedding layer

In traditional sequence recommendation models, the user and item are generally embedded with one-hot vectors by using user and item ID, whose dimension is equal to the size of the item set. However, this representation suffers from serious dimensional disaster and data sparsity problems [31], especially when the size of the item set reaches millions or even larger.

To address this issue, we consider user-provided reviews as a strong supplementary to understanding user behavior. Thus, RTiSR learns the user and item representation via exploiting reviews. Suppose that a review has m words. By employing a word embedding matrix $\mathbf{E}^w \in \mathbb{R}^{d_w \times |\mathcal{V}'|}$ with a table look-up operation, each review can be encoded as a matrix $\mathcal{D}'$:

$$\mathcal{D}' = [\mathbf{wd}_1; \mathbf{wd}_2; \ldots; \mathbf{wd}_m] \in \mathbb{R}^{d_w \times m} \tag{1}$$

where $\mathbf{wd}_i$ is the i-th word embedding in a review, d_w is the dimension of a word vector, and $\mathcal{V}'$ is the whole vocabulary of words. The pre-trained $\mathbf{E}^w$ can be obtained via word embedding methods such as word2vec [32] and GloVe [33], which are widely used in NLP.

Compared to the basic LSTM unit, BiLSTM unit can capture the both of syntax and meaning of words well, which contains the forward and backward LSTM units. Given the input word $\mathbf{w}_k$ and previous hidden state $\mathbf{h}_{k-1}$, the sequential updating process of a LSTM unit can be expressed as:

$$\mathbf{h}_k = \mathbf{LSTM}(\mathbf{h}_{k-1}, \mathbf{wd}_k); (1 \le k \le m) \tag{2}$$

Let $\overrightarrow{\mathbf{h}}_k$ and $\overleftarrow{\mathbf{h}}_k$ represent the hidden state of the forward and backward LSTMs at the k-th step, then the final hidden state of a BiLSTM unit can be formulated as:

$$\widetilde{\mathbf{h}}_k = \mathbf{BiLSTM}(\widetilde{\mathbf{h}}_{k-1}, \mathbf{wd}_k)(1 \le k \le m) \tag{3}$$

where $\widetilde{\mathbf{h}}_k$ is the final hidden state of a BiLSTM unit, i.e., $\widetilde{\mathbf{h}}_k = \left[\overrightarrow{\mathbf{h}}_k, \overleftarrow{\mathbf{h}}_k\right]$. Please note that consider the balance between efficiency and performance, we choose BiLSTM as our encoder. This function can be realized by other RNN networks, such BiGRU or Transformer.

Thus, the i-th review embedding vector $\mathbf{e}_i$ can be obtained:

$$\mathbf{e}_i = \widetilde{\mathbf{h}}_1 \oplus \widetilde{\mathbf{h}}_2 \oplus \cdots \oplus \widetilde{\mathbf{h}}_m \tag{4}$$

Time interval model. In real-world recommendation scenarios, the user's behavior sequence is complex. As shown in Fig. 4, the time intervals between two adjacent reviews can be various. Intuitively, the two actions within a short time interval tend to share a closer relationship than two actions within a long time interval. Thus this kind of temporal distance deserves special handling. However, previous time-aware sequential recommendation methods [11, 12] always assume that the items in sequence can be considered evenly spaced. The only influence factor to the following items is position.

In this work, we view the time interval between two consecutive behaviors as the relation between two interactions. To capture the influence of time intervals on review

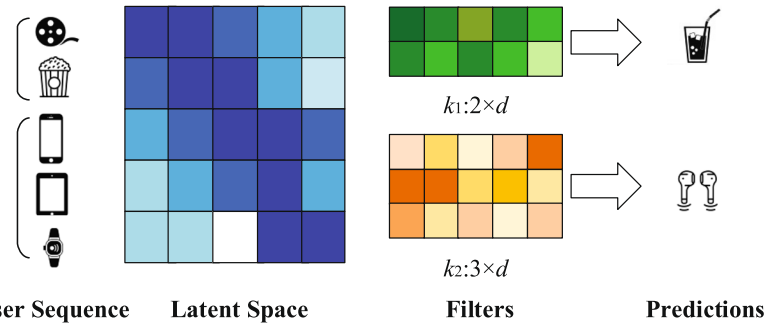


Fig. 6 Illustration of embedding image

embedding, we introduce a time interval model to assign dynamic weights to different reviews.

Let $T = \{t_1, t_2, \ldots, t_n\}$ denotes the timestamp sequence of user u or item v, the time interval of two items can be denoted as $|t_{i+1} - t_i|, 1 \leq i < n$. To obtain the personalized time interval, we divide it by the interval between the longest and the shortest time interval (except for 0) in an aspect-aware sequence. Specifically, for user u, the personalized interval $\bigtriangleup t_i^u$ is:

$$\bigtriangleup t_i^u = \frac{|t_{i+1}^u - t_i^u| - min(\bigtriangleup t^u)}{max(\bigtriangleup t^u) - min(\bigtriangleup t^u)} \tag{5}$$

where $min(\bigtriangleup t^u)$ and $max(\bigtriangleup t^u)$ is the minimum and maximum value of all time intervals in the user u's behavior sequence.

Thus, the final review embedding $\mathbf{e}_i'$ can be expressed as:

$$\mathbf{e}_i' = \mathbf{e}_i \times \frac{1}{\bigtriangleup t_i}. \tag{6}$$

The flexible sequence pattern learning layer

To capture the multiple sequence patterns hidden in the user behaviors, we introduce several convolution kernels $\mathbf{F}_*$ to obtain user and item latent features with a flexible order. First, RTiSR constructs the $L \times d_c$ (i.e., $d_c = d_l \times m$) matrix $\mathbf{E}^s$ as the embedding image of the previous n reviews of user u or item v in the sequence, which can be expressed as follow:

$$\mathbf{E^s} = [\mathbf{e}_1', \mathbf{e}_2', \ldots, \mathbf{e}_L'] \in \mathbb{R}^{L \times d_c} \tag{7}$$

As a result, the collective sequence patterns can be considered as the local features of this embedding image $\mathbf{E^s}$.

Then it utilizes a set of convolution filters with different size to search for sequential patterns. Figure 6 shows two horizontal filters that capture two union-level sequential patterns, which represented as $h \times d_c$ matrices. Specially, one horizontal filter $\mathbf{F}^1$

with size of $2 \times d_c$, while the other filter $\mathbf{F}^2$ with size of $3 \times d_c$. They search the hidden sequential patterns by sliding over the rows of $\mathbf{E}$. For instance, the recommended item "drinks" can be found, when adopting the first filter $\mathbf{F}^1$ works on the sequential pattern "(movie, popcorn)". Similarly, it picks up the sequential pattern "(iphone, ipad, iwatch) → airpods", since iphone, ipad and iwatch have large value in the latent dimensions via the second filter $\mathbf{F}^2$. More details are explained below.

In this layer, we employ several convolution kernels $\mathbf{F}_*$ with different size to learn the sequence dependencies with a flexible order. A set of n_c horizontal filters $\mathbf{F}^k \in \mathbb{R}^{h \times d_c}$ $(1 \leq k \leq n_c)$ is utilized to achieve it, where $h \in \{1, \ldots, L\}$ is the filter's height. For instance, if $L = 5$, one may choose to have $n_c = 10$ filters, two for each h in $\{1, 2, 3, 4, 5\}$. In detail, $\mathbf{F}^k$ slides from top to bottom on $\mathbf{E^s}$ and interacts with all horizontal dimensions of $\mathbf{E^s}$ of the reviews $i, 1 \leq i \leq L - h + 1$. Therefore, the i-th convolution value of filter $\mathbf{F}^k$ is:

$$c_i^k = \phi_c\left(\mathbf{F}^k \odot \mathbf{E^s}_{i:i+h-1}\right) \tag{8}$$

where $\mathbf{E^s}_{i:i+h-1}$ is the sub-matrix from the row i to row $(i - h + 1)$ of $\mathbf{E^s}$, ϕ_c is the activation function.

The final convolution result of $\mathbf{F}^k$ is:

$$\mathbf{c}^k = \left[c_1^k, c_2^k, \ldots, c_{n-h+1}^k\right] \tag{9}$$

After that, the max operation is introduced to extract the maximum value from (9), where it represents the most meaningful feature extracted by filter $\mathbf{F}^k$. For the n_c convolution filters, the output value $\mathbf{x} \in \mathbb{R}^{n_c}$ is:

$$\mathbf{z} = \left\{max(\mathbf{c}_1), max(\mathbf{c}_2), \ldots, max(\mathbf{c}_{n_c})\right\} \tag{10}$$

Finally, the user and item representations can be obtain by utilizing a fully connected layer on the output of convolutional layer:

$$\mathbf{o} = ReLU\left(\mathbf{W}_f' \times \mathbf{z} + \mathbf{b}_f'\right) \tag{11}$$

where $\mathbf{o} \in \mathbb{R}^{d_o}$ is the latent vector of a user or an item with the size of d_o.

In the flexible sequence pattern learning layer, the horizontal convolution filter interacts with every successive h review in the embedding matrix E^s. With sliding horizontal filters of varied heights, a significant signal will be extracted from the review sequence including multiple sequence patterns, user preference, and item features with position and time interval information.

Here we put the user and item representations (i.e., $\mathbf{o}^u$, $\mathbf{o}^v$) in the output of the full-connected layer for two main reasons: (1) it can have the ability to generalize other recommendation models. (2) Our model parameters can be initialization with other generalized models' well-trained parameters. As stated in [34], such pre-training is vital to model performance.

The prediction layer

After the flexible sequence pattern learning layer, we obtain the combined representation of review, time interval, and sequence patterns to user u and item v. To estimate the likelihood of next purchase item, we utilize a factorization-based model [35] to estimate the user preference score of item v. The model details can be expressed as follows:

$$\hat{y}_v^u := \omega_0 + \sum_{i=1}^{d_r} \omega_i \Psi_i + \sum_{i=1}^{d_r} \sum_{j=i+1}^{d_r} \left\langle v_i, v_j \right\rangle \Psi_i \Psi_j \tag{12}$$

where $\mathbf{\Psi}$ is the concatenate result of $\mathbf{o}^u$ and $\mathbf{o}^v$ (i.e., $\mathbf{\Psi} = \mathbf{o}^u \oplus \mathbf{o}^v$), ω_* are the bias terms, $\left\langle \mathbf{v}_i, \mathbf{v}_j \right\rangle$ represents the interaction between the i-th and j-th variables, $d_r = 2d_o$, k is the size of variable $\mathbf{v}$.

Model optimization

The model parameters of RTiSR include review embedding, two hybrid neural networks, and factorization machine. For each user-item pair (u, v) , we extract the L successive interactions of user u and item v as a training instance $l = (S^{u,L}, S^{v,L})$. Following [20], for each training instance l_v with target item v, we randomly sample ζ negative items (i.e., $v' \notin S^u$) with their L sucessive interactions, denoted as $\mathcal{N}(v)$. Let C^u be the collection of user u's all training instances. We transform model output scores into the range (0, 1) by a sigmoid function $\sigma(x) = \dfrac{1}{1+e^{-x}}$ and adopt binary cross-entropy as the loss function:

$$\mathcal{L} = \sum_u \sum_{l_v \in C^u} \left(-log(\sigma(\hat{y}_v^u)) + \sum_{j \in \mathcal{N}(v)} -log(1 - \sigma(\hat{y}_j^u)) \right) + \lambda \parallel \Theta \parallel_F^2 \tag{13}$$

$\Theta = \{\mathbf{W}_*, \mathbf{U}_*, \mathbf{b}_* \mathbf{F}, \mathbf{W}', \mathbf{b}', \mathbf{V}, \boldsymbol{\omega}_*\}$ denotes all model parameters, λ is the regularization term. $\| \cdot \|_F$ is the Frobenius norm. We adopt the Adam optimizer [36] to optimize our model. Meanwhile, the mini-batch SGD is applied to speed up the training efficiency. To avoid the over-fitting, *dropout* regularization method is used to the full-connect layer.

Algorithm 1: The recommendation process of RTiSR

Input: the user set $\mathbf{U}$ with each user's latest L interaction sequence $S^{(u,L)}$ and time sequence T^u, the candidate item set Γ^u with each item v's latest L interaction sequence $S^{(v,L)}$ and time sequence T^v

Output: Recommendation list $R(u)$ for each user

```
foreach u ∈ U do
    Y^u ← ∅ ;
    o^u ← UNet(S^(u,L), T^u) ;
    foreach v ∈ Γ^u do
        o^v ← INet(S^(v,L), T^v) ;
        ŷ_v^u ← FM(o^u, o^v);
        Y^u ← Y^u ⋃ {ŷ_v^u} ;
    end
    R(u) ← Top( Y^u, K ) ;
end
```

Algorithm 2: U-Net

Input: each L interaction sequence $S^{u,L}$ of user u, and the corresponding time sequence T^u
Output: user representation $\mathbf{o}^u$

$\mathbf{E}_s^u \leftarrow \varnothing$;
foreach $r_i^u \in \mathcal{S}^{(u,L)}$ **do**
  $\mathcal{D}_i^u \leftarrow$ **Encoder**(r_i^u);
  $\mathbf{e}_i^u \leftarrow$ **BiLSTM**$(\mathcal{D}_i^u)$;
  $\triangle t_i^u \leftarrow$ **Encoder**(t_i^u, t_{i+1}^u);
  $\mathbf{e}'^u_i \leftarrow \mathbf{e}_i^u \times \frac{1}{\triangle t_i^u}$;
  $\mathbf{E}_s^u \leftarrow \mathbf{E}_s^u \bigcup \{\mathbf{e}'^u_i\}$;
end
foreach $F^k \in F^*$ **do**
  $c_k^u \leftarrow$ **Conv**$(F^k, \mathbf{E}_s^u)$;
  $c^u \leftarrow$ **Insert**(c_k^u);
end
$\mathbf{x}^u \leftarrow$**Max**(c^u);
$o^u \leftarrow$ **MLP**$(\mathbf{x}^u)$;
Result: o^u

In the recommendation phase, we take u's latest L interactions $S^{u,L}$, and the corresponding candidate item v with their latest L interactions $S^{v,L}$ as input. We obtain the predict score of item v ($\hat{y}_v^u$) via calling functions **UNet**, **INet** and **FM**, and then pick the K items with the highest scores as recommendation list. The implementation details of U-Net and RTiSR are shown in Algorithm 1 and Algorithm 2. The computation complexity of making recommendation to all user is $O(|U||\Gamma|(L \times T_{BiLstm} + n_c \times T_{Conv}))$, where the complexity of BiLSTM and Conv operation are T_{BiLstm} and T_{Conv}.

Experiments

To comprehensively evaluate the performance of RTiSR, we conduct a set of experiments to answer the following research questions:

- **RQ1**: How does RTiSR perform as compared with state-of-the-art sequence and review based recommendation models;
- **RQ2**: What is the influence of aspect reviews, time interval aware embedding and convolutional layer in RTiSR;
- **RQ3**: How do the key hyperparameters affect the performance of RTiSR, such as the dimension (d_o) of user/item latent factors, the height size (h) of convolution filter.

Experimental settings

Dataset: We evaluate RTiSR on five public datasets with different characteristics. The four datasets from the sub-category of Amazon[1] including **Musical Instruments** (MIs), **Automotive** (Auto), **Luxury Beauty** (LB), and **Beer**, while dataset the **Yelp**[2] from Yelp Challenge 2019 contains data from Jan. 1, 2019, to Aug. 31st, 2019. Moreover, we filter all datasets that all users and items have at least 20 interactions. The basic statistics of each dataset are described in Table 2.

[1] http://jmcauley.ucsd.edu/data/amazon.
[2] https://www.yelp.com/dataset.

Table 2 The statistics of datasets

Datasets	# users	# items	# reviews	Density (%)
Yelp	366,715	60,785	1,569,264	0.007
Beer	40,213	110,419	2,924,127	0.066
LB	19,947	1798	23,799	0.066
Auto	2928	1835	20,473	0.381
MIs	1429	900	10,261	0.798

Baselines: To demonstrate the effectiveness, we compare our proposed RTiSR with the following methods:

- **BPRMF** [37]: It is Matrix Factorization-based ranking algorithm that optimizes the pairwise ranking loss with implicit feedback. It is a popular baseline for item recommendation.
- **DeepCoNN** [27]: This is a state-of-the-art review-based recommendation method, which leverages convolution neural network to jointly model the users and items from reviews.
- **SLRC** [38]: It introduces Hawkes Process into Collaborative Filtering (CF), and explicitly addresses two item-specific temporal dynamics: short-term effects and life-time effects.
- **CFKG** [39]: It is a knowledge-based representation learning approach that embeds heterogeneous entities for personalized recommendation, which incorporates the defined user-item knowledge-graph structure to improve the recommendation performance.
- **CORE** [40]: It is a sequential recommendation method with a representation consistency encoder for representing sequence embeddings and item embeddings in the same space. This method is the state-of-the-art baseline for sequential recommendation.
- **SINE** [41]: It is a state-of-the-art sequential recommendation method that simultaneously considers multiple interests of a user and aggregates them to predict the user's current intention.
- **LightSANs** [42]: It is a novel transformer-based sequential recommendation, which introduces the low-rank decomposed self-attention, which projects the user's historical items into a small constant number of latent interests and leverages item-to-interest interaction to generate the context-aware representation.

Evaluation metrics: To evaluate the performance, we adopt two well-known metrics in Top-N recommendation: *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) [31]. In detail, HR is calculated as:

$$HR@K = \frac{Number\ of\ Hits@K}{|GT|} \tag{14}$$

where a hit is defined as a test item appears in the recommendation list. GT denotes the ground-truth item set.

Compared to the recall-based metric HR, NDCG is a measure of ranking quality, where positions are discounted logarithmically. It accounts for the position of the hit by assigning higher scores to hits at the top ranks:

$$NDCG@K = Z_K \sum_{i=1}^{K} \frac{2^{r_i} - 1}{log_2(i+1)} \tag{15}$$

where Z_K is the normalization term. r_i is a binary value, where $r_i = 1$ if the item is in the test, otherwise $r_i = 0$. The larger values of HR and NDCG indicate better performance the model has. In the evaluation, we report the average value of both metrics as the final score of each user.

Implementation details: Following [17, 34, 43], we adopt the *leave-one-out* evaluation method to testing the above mentioned models. For user u, we leverage the latest interaction $S^u_{|S^u|}$ as the test set, and the penultimate interaction $S^u_{|S^u|-1}$ as the validation set, while the remaining interactions are used for training. Specifically, we randomly sample 100 items that are not interacted with by user u, while ranking the test item among the 100 items. To trade-off the performance and efficiency of RTiSR, the Adam optimizer is configuration with learning rate tuned in [0.0001, 0.0005, 0.001, 0.005] . The batch size is tuned in [32, 64, 128, 256] . Furthermore, we also tune the dimension of word embedding with the range of [100, 200, 300, 400, 500]. Moreover, the length of succesive order L is form $\{3, \ldots, 9\}$, and the height h of horizaontial filters is from $\{1, \ldots, L\}$. For each height h, the number of horizontal filters n_c is from $\{4, 8, 12, 16, 20, 24, 28, 32\}$. Note that we tune hyperparameters using the validation set. For a fair comparison, we reused the hyper-parameters of all mentioned models as reported. Otherwise, we carefully tune them to ensure that they achieve the best performance. We implement our RTiSR model in *Pytorch*[3]

All experiments are conducted on a tower server with configuration of Intel(R) Xeon(R) CPU E5-2680 v3 @2.50 GHz, RAM 128 GB. The operating system is Ubuntu 20.04.3 LTS. Meanwhile, JDK 1.8, Python 3.6, and PyTorch 1.8 are also installed.

Overall performance (RQ1)

We first evaluate the overall recommendation performance of all mentioned models on different datasets. The comparison results with varying the length of recommendation list (K) are shown in Table 3. To better understand these comparison results, the Friedman test [44] with the win/loss counts are also reported. These results are shown in the last two rows of Table 3. From the table, we have the following observations:

- RTiSR consistently outperforms most baselines on all datasets, which achieves the best performance and the highest F-rank value. Compared to the strongest baselines on HR or NDCG, RTiSR still achieves different degrees of improvement on the five datasets. In detail, RTiSR average outperforms the strongest baselines by 8.4%, 4.7%, 8.1%, 3.7%, 9.0% on dataset MIs, Auto, LB, Beer, Yelp, respectively. It demonstrates the effectiveness of RTiSR, which is attributed to better capturing the union-level

[3] https://pytorch.org/.

Table 3 Performance comparison of different methods for item recommendation task on five datasets

Dataset	Metric	BPRMF	CORE	SINE	LightSANs	SLRC	DeppCoNN	CFKG	RTiSR	*Impv.*	*p*-value
MIs	HR@5	0.4033	**0.5976**	0.5153	0.5477	0.5213	0.4136	0.5250	0.5951	–	–
	NDCG@5	0.2921	0.3942	0.3957	0.4078	0.3890	0.2923	0.3625	**0.4207**	3.16%	8.9e−2
	HR@10	0.4551	0.5869	0.5612	0.5698	0.5890	0.4741	0.5548	**0.6874**	16.71%	1.1e−4*
	NDCG@10	0.3007	0.4056	0.4034	0.4248	0.4038	0.3136	0.3890	**0.4569**	7.56%	4.9e−2*
	HR@20	0.4568	0.5935	0.5655	0.5786	0.5992	0.4947	0.5660	**0.6957**	16.10%	8.2e−4*
	NDCG@20	0.3215	0.4104	0.4099	0.4292	0.4127	0.3531	0.3977	**0.4599**	7.15%	4.2e−2*
Auto	HR@5	0.4417	0.6097	0.5559	0.6068	0.5023	0.4319	0.5501	**0.6207**	1.80%	1.3e−2*
	NDCG@5	0.2721	0.4660	0.4146	**0.4703**	0.3140	0.2812	0.3007	0.3861	–	–
	HR@10	0.4668	0.6107	0.5663	0.6456	0.5662	0.4638	0.5945	**0.7039**	9.03%	4.9e−2*
	NDCG@10	0.3102	0.4448	0.4200	**0.4874**	0.3871	0.3008	0.3915	0.4833	–	–
	HR@20	0.4725	0.6187	0.5925	0.6591	0.5810	0.4850	0.6110	**0.7099**	7.71%	7.2e−2
	NDCG@20	0.3320	0.4471	0.3274	0.3842	0.4020	0.3122	0.4201	**0.4929**	10.24%	4.8e−2*
LB	HR@5	0.4359	0.5269	0.5311	0.4973	0.4903	0.4759	0.4779	**0.5531**	4.14%	7.5e−3*
	NDCG@5	0.2805	0.3639	0.3295	0.3538	0.3656	0.3032	0.3137	**0.3992**	9.19%	6.1e−4*
	HR@10	0.4620	0.5279	0.5347	0.5016	0.5243	0.4904	0.5021	**0.6075**	13.62%	2.0e−3*
	NDCG@10	0.2917	0.3791	0.3362	0.3581	0.3907	0.3426	0.3383	**0.4251**	8.80%	2.0e−2*
	HR@20	0.4801	0.5323	0.5404	0.5189	0.5309	0.5011	0.5201	**0.6089**	12.68%	3.1e−3*
	NDCG@20	0.3382	**0.4421**	0.3434	0.3618	0.4102	0.3213	0.3566	0.4315	–	–
Beer	HR@5	0.3625	0.4938	0.4611	0.5173	0.4526	0.3820	0.3661	**0.5617**	8.58%	4.1e−2*
	NDCG@5	0.2496	0.3624	0.3633	0.3558	0.2926	0.2817	0.2457	**0.3715**	2.26%	2.7e−3
	HR@10	0.3998	0.5793	0.5673	0.5769	0.4725	0.3947	0.3870	**0.5921**	2.21%	7.3e−3*
	NDCG@10	0.2651	0.3963	0.3706	0.3616	0.3322	0.2920	0.2681	**0.4107**	3.63%	0.17
	HR@20	0.4195	0.5868	**0.5908**	0.5838	0.5026	0.4218	0.4102	0.5026	–	–
	NDCG@20	0.3005	0.3556	0.3987	0.3348	0.3539	0.3198	0.3108	**0.4211**	5.62%	0.18

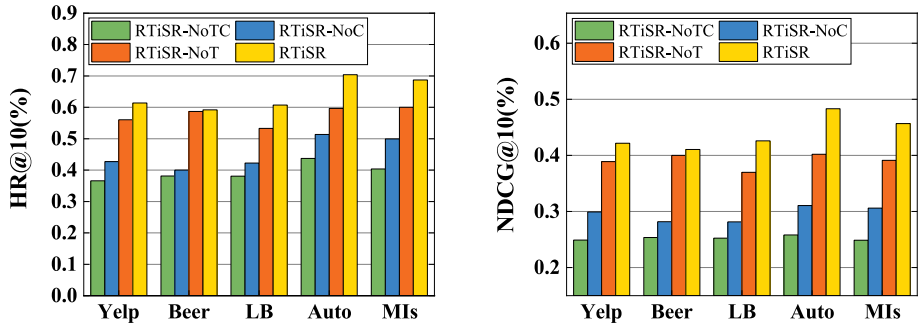
Table 3 (continued)

Dataset	Metric	BPRMF	CORE	SINE	LightSANs	SLRC	DeppCoNN	CFKG	RTiSR	*Impv.*	*p*-value
Yelp	HR@5	0.4331	0.4868	0.5166	0.5326	0.4903	0.4741	0.4726	**0.5637**	5.84%	3.9e−2*
	NDCG@5	0.2521	0.3601	0.3492	0.3439	0.3107	0.3112	0.2635	**0.3826**	6.25%	3.6e−2*
	HR@10	0.4537	0.5159	0.5227	0.5411	0.5130	0.5001	0.4807	**0.6139**	13.45%	1.2e−2*
	NDCG@10	0.2813	0.3956	0.3682	0.3457	0.3261	0.3138	0.2840	**0.4217**	6.60%	0.10
	HR@20	0.4807	0.5338	0.5556	0.5469	0.5337	0.5238	0.5010	**0.6219**	11.93%	3.8e−3*
	NDCG@20	0.3101	0.3964	0.3685	0.3514	0.3740	0.3329	0.3124	**0.4361**	10.02%	4.5e−2*
Statistic	Win/Loss	(18/0)	(17/1)	(17/1)	(16/2)	(18/0)	(18/0)	(18/0)			
	F-rank[a]	1.33	6.23	5.13	5.67	4.65	2.30	3.00	**7.68**		

We use bold and underline fonts to denote the best performance and second best performance method in each metric respectively. *Impv.* refers to the relative improvement of RTiSR over the corresponding baseline, and *p*-value. Measures the significance of RTiSR relative to baseline improvement

[a] A higher F-rank value indicates a higher recommendation performance

* denotes that the corresponding improvement has passed the significant test at the significance level of 0.05



(a) HR@10 (b) NDCG@10

Fig. 7 Performance comparison of different components of RTiSR

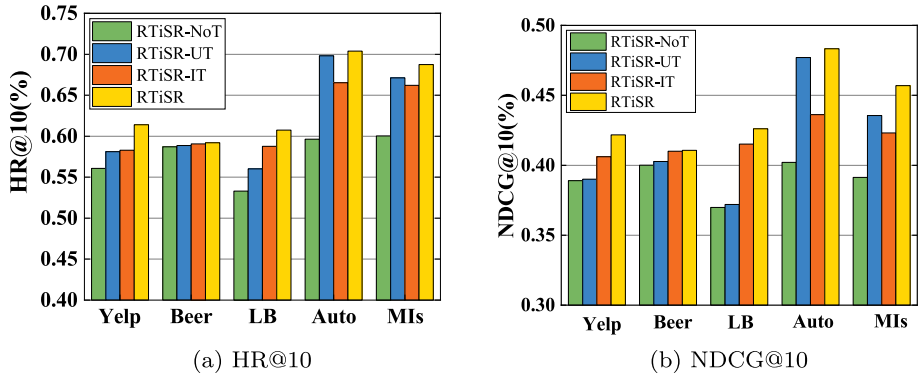
sequential dependencies with a flexible order. Moreover, the superior performance of RTiSR reflects the rationality of utilizing user reviews and exact time interval information in improving the recommendation performance.

- In most conditions, sequential recommendation methods like CORE and TiSASRec SINE perform better than the CF-based methods (i.e., BPRMF, CFKG) and review-based methods (i.e., DeppCoNN). The main reason stands for the significant performance gap: although DeppCoNN and CFKG utilize the user review and knowledge-graph to enhance the user and item representation, the capacity of BPRMF, DeppCoNN, and CFKG are limited in modeling user preference without considering the sequential dependencies hidden in the user behaviors.
- Our proposed method shows significant improvement (p-value $\leqslant 0.05$) on all datasets compared to the SOTA baseline. The reasons for this performance gap are (1) compared with CF-based methods (e.g., CFKG, and SLRC), RTiSR can capture users' dynamic preferences by modeling users' sequential patterns in historical interactions. (2) compared with sequential recommendation models (e.g., CORE, SINE, and LightSANs), RTiSR introduces the aspect review information to improve the user and item embedding quality. Based on it, RTiSR regards the embedded sequential reviews with explicit time interval information as an image, then employs multi-size convolution fitters to capture the collective sequential dependencies with flexible order, rather than the point-wise way. (3) Compared with the review-based recommendation method (i.e., DeepCoNN), RTiSR improves the performance of sequence recommendation by considering time interval information to assign dynamical weights to different reviews in aspect-aware review sequence.

Ablation study (RQ2)

RTiSR introduces two essential extensions for the sequential recommendation to capture the user and item representations. In this section, we conduct a set of ablation experiments to analyze different components' impacts.

Analysis of different components. To analyze the influence of the time interval aware model and convolution operations on RTiSR, we compare the RTiSR with the following variants:



(a) HR@10 (b) NDCG@10

Fig. 8 Performance comparison of different time components of RTiSR

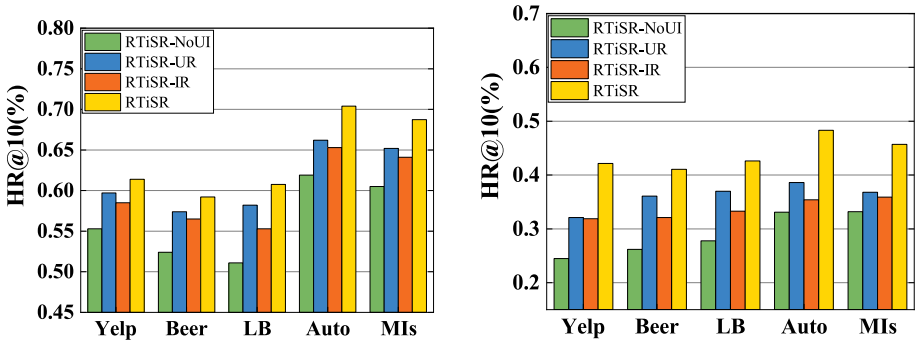
- **RTiSR-NoTC**: Both time interval model and convolution filter are not used in this variant.
- **RTiSR-NoC**: It only uses the time interval model to enhance the user/item representation.
- **RTiSR-NoT**: It only integrates the convolution filter into the RTiSR framework.

The performance of RTiSR and its three variants on five datasets are shown in Fig. 7. We can observe that incorporating a time interval model can improve the recommendation performance significantly (i.e., RTiSR-NoC performs better than RTiSR-NoTC). However, compared to the variant RTiSR-NoC, RTiSR-NoT achieves a better performance on all evaluation metrics (i.e., HR@10 and NDCG@10) over five datasets, except for RTiSR. It indicates the necessity of learning the collective dependencies flexibly in the sequential recommendation. Finally, by incorporating these two parts, the complete model RTiSR outperforms its three variations on all evaluation matrices. Due to the limited space, we only show the results on HR and NDCG for top-10 recommendations, and other metrics show similar results.

Analysis of different time intervals. In RTiSR, the time interval information is introduced to guide RTiSR for capturing the temporal pattern in user sequential behaviors, which considers the impact from the user and item perspective. Thus, we also evaluate the effect of the aspect-aware time interval model on the performance of RTiSR. We introduce two variants:

- **RTiSR-UT**: We introduce the time interval-aware model in the U-Net, which is equivalent to only considering the temporal pattern of the user.
- **RTiSR-IT**: Instead of introducing a time interval-aware model for the U-Net, the variant utilizes it on I-Net to explore the temporal pattern of item.

Figure 8 shows the performance of RTiSR with its three variants on five datasets. We can observe that the recommendation performance has a significant improvement, when inserting the time interval information into U-Net or I-Net. In detail, RTiSR-UT performs better than RTiSR-IT on HR and NDCG over dataset MIs and



(a) HR@10 (b) NDCG@10

Fig. 9 Performance comparison of different aspect components of RTiSR

Auto, while RTiSR-IT shows better performance on dataset LB, Beer, and Yelp. Such a difference in performance improvement is due to the nature of the five datasets. The data sparsity of MIs and Auto is much better than that of the other three datasets, indicating that more interactions can be supported to capture the sequence dependencies. Thus, the complete model RTiSR performs better with both time interval models.

Analysis of different aspect reviews. In RTiSR, the user-aspect and item-aspect reviews are used to infer more effective embeddings. To validate the importance of different aspect reviews on the recommendation performance, We introduce three variants:

- **RTiSR-NoUI**: Both time user-side model and item-side model are not used in this variant, the user/item representations are generated from an MLP, without any aspect review information embedded.
- **RTiSR-UR**: It only uses the user-side model to generate the user representations, which contain the user-aspect review information.
- **RTiSR-IR**: It only uses the item-side model to generate the item representations, which contain the item-aspect review information.

Fig. 9 shows the performance of RTiSR with its three variants on five datasets. We can observe that the recommendation performance has a significant improvement, when embedding user/item-aspect information in the representation vector. In detail, RTiSR-UT performs better than RTiSR-IT, which is attributed to the user-aspect reviews containing more dynamic preference information, while the item-aspect review is more about itself. Moreover, we observed that aspect review information improves even more on sparse datasets (e.g., Yelp, Beer, and LB), which shows that when there is a lack of effective interactive data, the reviews can provide the recommender model with fine-grained information for refining user and item embeddings for improving the accuracy of recommendation.

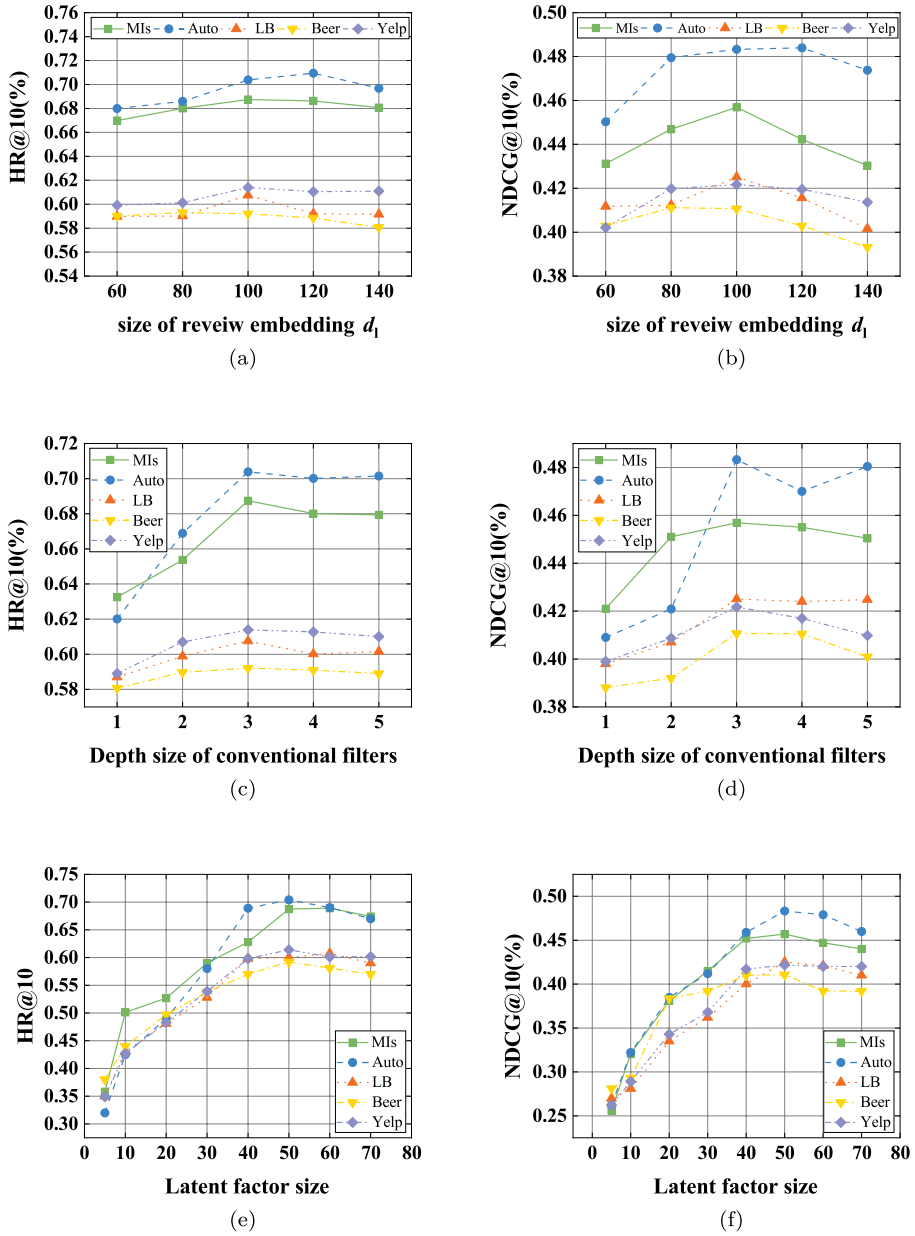


Fig. 10 Performance as a result of varying key parameters. Performance as a result of RTiSR with varying d_l, h and d_o

Study of RTiSR (RQ3)

As the time interval aware convolutional layer plays a pivotal role in RTiSR, we investigate its impact on the performance. In this section, we start by learning the influence of review embedding size d_l (i.e., the input of this layer). We then study how the incremental depth size ($\bigtriangleup h$) of the convolution filters affects the performance. Furthermore, we analyze the influences of latent factor size (i.e., the output of this layer).

Effect of review embedding size d_l. To investigate whether RTiSR can benefit from user-provided reviews, we vary the size of review embedding. Specifically, we search

the review embedding sizes in the range of {60, 80, 100, 120, 140}. Figure 10a, b show the experiment results. We have the following observations:

- Increasing the size of review embedding substantially improves the recommendation performance on all datasets in terms of HR@10 and NDCG@10. In detail, RTiSR consistently improves dense dataset MIs and Auto over the other three datasets. We attribute the improvement to the effective exploration of user reviews for user and item representations: reviews contain valuable sentiment information about user preference and item features. Thus, RTiSR can significantly enhance the performance of sequential recommendations with user reviews.
- When further increasing the size of review embedding with larger than 100, we find that it leads to overfitting over all datasets. This might be caused by applying a larger review embedding size that might introduce noises to the representation learning. Thus, it verifies that setting $d_l = 100$ is sufficient to represent reviews.

Effect of convolution fitter with varied depth size h. We vary h to explore how much of RTiSR can gain from the collective sequential dependencies with a flexible orders, while keeping other optimal hyper parameters unchanged. We set $n_c = 20$. $h = 3$ denotes the RTiSR employs a set of conventional filter to study the effect of collective sequential dependencies with successive 3 interactions. Figure 10c, d show the experiments result on HR and NDCG respectively. We can find that increasing h substantially improves the recommendation performance on all datasets. On the dense dataset MIs and Auto, RTiSR utilizes the extra information provided by a larger h, and $h = 3$ performs the best, suggesting the benefits of learning the sequential dependencies in a collective-order. However, RTiSR does not consistently benefit from a larger h. This is reasonable, since it introduces extra information while more noises.

Effect of latent factor number. Finally, we study the effect of user/item representation with different sizes on recommendation performance. Specifically, we vary the size of latent factor in the range of {5, 10, 20, 30, 40, 50, 60, 70}. Figure 10e, f show the experiment results on HR@10 and NDCG@10, respectively. We can observe that increasing the size of the latent factor substantially enhances the recommendation performance on all datasets. Clearly, on the dense MIs and Auto dataset, RTiSR with a larger latent factor size achieves consistent improvement over the other sparser dataset. We attribute the improvement to the effective user and item representation with a large latent factor size. The best performance of RTiSR can be achieved when the latent factor size is 50. Furthermore, on the dense dataset MIs and Auto, RTiSR with a larger latent factor size achieves consistent improvement over the other sparser dataset, contributing to the more extra information brought by the dense datasets.

Conclusion

In this paper, we propose a review-driven time interval aware neural network for sequential recommendation, which captures the aspect-aware sequence dependencies from exploiting reviews. We notice that a few studies attempt to enhance the performance of sequential recommendation with capturing the temporal dynamical of item-aspect reviews. On this basis, we view the user-provided reviews set as the aspect-aware review sequence, then introduce a time interval to assign dynamical weights to different reviews in the aspect-aware sequence. We leverage a hybrid neural network with combined of

BiLSTM and CNN to exploit the collective sequence dependencies with a flexible order from user-aspect and item-aspect review sequences respectively. Based on these methods, it makes RTiSR more suitable to capture the dynamic changes of user preference and item features. Finally, extensive experiments are conducted to test the RTiSR performance. The experimental results show the effectiveness and superiority of RTiSR in terms of HR and NDCG, compared to several SOTA models consistently.

Acknowledgements
This work is supported in part by the National Natural Science Foundation of China under Grants 62072429, in part by the Chinese Academy of Sciences "Light of West China" Program, and in part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008, HZ2021017), and the "Fertilizer Robot" project of Chongqing Committee on Agriculture and Rural Affairs.

Author contributions
All authors contributed to developing the ideas and writing and reviewing this manuscript. All authors read and approved the submitted manuscript.

Availability of data and materials
The dataset has no restrictions that all the data can be acquired in the related site.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 27 July 2022 Accepted: 23 February 2023
Published online: 12 March 2023

References

1. Resnick P, Varian HR. Recommender systems. Commun ACM. 1997;40(3):56–8.
2. Roy D, Dutta M. A systematic review and research perspective on recommender systems. J Big Data. 2022;9(1):1–36.
3. Linden G, Smith B, York J. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. 2003;7(1):76–80.
4. Widiyaningtyas T, Hidayah I, Adji TB. User profile correlation-based similarity (UPCSIM) algorithm in movie recommendation system. J Big Data. 2021;8(1):1–21.
5. Hidasi B, Tikk D. General factorization framework for context-aware recommendations. Data Min Knowl Disc. 2016;30(2):342–71.
6. Chen X, Xu H, Zhang Y, Tang J, Cao Y, Qin Z, Zha H. Sequential recommendation with user memory networks. In: Proceedings of the 11th ACM international conference on web search and data mining; 2018. p. 108–16.
7. Wang D, Zhang X, Xiang Z, Yu D, Xu G, Deng S. Sequential recommendation based on multivariate Hawkes process embedding with attention. IEEE Trans Cybern. 2021;2021:1.
8. Feng S, Li X, Zeng Y, Cong G, Chee YM, Yuan Q. Personalized ranking metric embedding for next new poi recommendation. In: 24th international joint conference on artificial intelligence; 2015.
9. Rendle S, Freudenthaler C, Schmidt-Thieme L. Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on world wide web; 2010. p. 811–20.
10. Wu C-Y, Ahmed A, Beutel A, Smola AJ, Jing H. Recurrent recommender networks. In: Proceedings of the 10th ACM international conference on web search and data mining; 2017. p. 495–503.
11. Ying H, Zhuang F, Zhang F, Liu Y, Wu J. Sequential recommender system based on hierarchical attention networks. In: 27th international joint conference on artificial intelligence IJCAI-18; 2018.
12. Li J, Wang Y, Mcauley J. Time interval aware self-attention for sequential recommendation. In: WSDM'20: the 13th ACM international conference on web search and data mining; 2020.
13. Garcin F, Dimitrakakis C, Faltings B. Personalized news recommendation with context trees. In: Proceedings of the 7th ACM conference on recommender systems; 2013. p. 105–12.
14. He R, McAuley J. Fusing similarity models with Markov chains for sparse sequential recommendation. In: 2016 IEEE 16th international conference on data mining (ICDM). New York: IEEE; 2016. p. 191–200.
15. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks. Preprint; 2015. arXiv:1511.06939.
16. Quadrana M, Karatzoglou A, Hidasi B, Cremonesi P. Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the 11th ACM conference on recommender systems; 2017. p. 130–7.

17. Kang W-C, McAuley J. Self-attentive sequential recommendation. In: 2018 IEEE international conference on data mining (ICDM). New York. IEEE; 2018. p. 197–206.
18. Yuan W, Wang H, Yu X, Liu N, Li Z. Attention-based context-aware sequential recommendation model. Inf Sci. 2020;510:122–34.
19. Tang J, Wang K. Personalized top-*n* sequential recommendation via convolutional sequence embedding; 2018. p. 565–73. https://doi.org/10.1145/3159652.3159656.
20. Li C, Niu X, Luo X, Chen Z, Quan C. A review-driven neural model for sequential recommendation. In: Proceedings of the 28th international joint conference on artificial intelligence; 2019. p. 2866–72.
21. Liu Y, Zhang Y, Zhang X. An end-to-end review-based aspect-level neural model for sequential recommendation. Discrete Dyn Nat Soc. 2021;2021:1.
22. Gao J, Lin Y, Wang Y, Wang X, Yang Z, He Y, Chu X. Set-sequence-graph: a multi-view approach towards exploiting reviews for recommendation. In: Proceedings of the 29th ACM international conference on information and knowledge management; 2020. p. 395–404.
23. Dong X, Ni J, Cheng W, Chen Z, Zong B, Song D, Liu Y, Chen H, De Melo G. Asymmetrical hierarchical networks with attentive interactions for interpretable review-based recommendation. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34; 2020. p. 7667–74.
24. McAuley J, Leskovec J. Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM conference on recommender systems; 2013. p. 165–72.
25. Bao Y, Fang H, Zhang J. Topicmf: simultaneously exploiting ratings and reviews for recommendation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 28; 2014.
26. Tan Y, Zhang M, Liu Y, Ma S. Rating-boosted latent topics: understanding users and items with ratings and reviews. In: IJCAI, vol. 16; 2016. p. 2640–6.
27. Zheng L, Noroozi V, Yu PS. Joint deep modeling of users and items using reviews for recommendation. In: Proceedings of the 10th ACM international conference on web search and data mining; 2017. p. 425–34.
28. Catherine R, Cohen W. Transnets: learning to transform for recommendation. In: Proceedings of the 11th ACM conference on recommender systems; 2017. p. 288–96.
29. Tay Y, Luu AT, Hui SC. Multi-pointer co-attention networks for recommendation. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining; 2018. p. 2309–18.
30. Wu L, Quan C, Li C, Wang Q, Zheng B, Luo X. A context-aware user-item representation learning for item recommendation. ACM Trans Inf Syst (TOIS). 2019;37(2):1–29.
31. He X, Chen T, Kan M-Y, Chen X. Trirank: review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM international on conference on information and knowledge management; 2015. p. 1661–70.
32. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems; 2013. p. 3111–9.
33. Pennington J, Socher R, Manning CD. Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014. p. 1532–43.
34. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S. Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web; 2017. p. 173–82.
35. Rendle S. Factorization machines. In: 2010 IEEE international conference on data mining. New York: IEEE; 2010. p. 995–1000.
36. Kingma DP, Ba J. Adam: a method for stochastic optimization. In: ICLR (Poster); 2015.
37. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th conference on uncertainty in artificial intelligence; 2009. p. 452–61.
38. Wang C, Zhang M, Ma W, Liu Y, Ma S. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In: The world wide web conference; 2019.
39. Zhang Y, Ai Q, Chen X, Wang P. Learning over knowledge-base embeddings for recommendation. Preprint; 2018. arXiv:1803.06540.
40. Hou Y, Hu B, Zhang Z, Zhao WX. Core: simple and effective session-based recommendation within consistent representation space. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval; 2022. p. 1796–801.
41. Tan Q, Zhang J, Yao J, Liu N, Zhou J, Yang H, Hu X. Sparse-interest network for sequential recommendation. In: Proceedings of the 14th ACM international conference on web search and data mining; 2021. p. 598–606.
42. Fan X, Liu Z, Lian J, Zhao WX, Xie X, Wen J-R. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval; 2021. p. 1733–7.
43. Bayer I, He X, Kanagal B, Rendle S. A generic coordinate descent framework for learning from implicit feedback. In: Proceedings of the 26th international conference on world wide web; 2017. p. 1341–50.
44. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006;7(Jan.):1–30.

Publisher's Note