Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation

Zeping Yu[1], **Jianxun Lian**[2*], **Ahmad Mahmoody**[3], **Gongshen Liu**[1*] and **Xing Xie**[2]
[1]Shanghai Jiao Tong University
[2]Microsoft Research Asia
[3]Microsoft Corporation
zepingyu@foxmail.com, {jianxun.lian, xing.xie, ahmahmoo}@microsoft.com, lgshen@sjtu.edu.cn

Abstract

User modeling is an essential task for online recommender systems. In the past few decades, collaborative filtering (CF) techniques have been well studied to model users' long term preferences. Recently, recurrent neural networks (RNN) have shown a great advantage in modeling users' short term preference. A natural way to improve the recommender is to combine both long-term and short-term modeling. Previous approaches neglect the importance of dynamically integrating these two user modeling paradigms. Moreover, users' behaviors are much more complex than sentences in language modeling or images in visual computing, thus the classical structures of RNN such as Long Short-Term Memory (LSTM) need to be upgraded for better user modeling. In this paper, we improve the traditional RNN structure by proposing a time-aware controller and a content-aware controller, so that contextual information can be well considered to control the state transition. We further propose an attention-based framework to combine users' long-term and short-term preferences, thus users' representation can be generated adaptively according to the specific context. We conduct extensive experiments on both public and industrial datasets. The results demonstrate that our proposed method outperforms several state-of-art methods consistently.

1 Introduction

In the era of information overload, Recommender Systems (RS) are essential for online services and are broadly utilized in a variety of areas such as online shopping, movies, music and news reading services. Two paradigms of recommender systems are most popular nowadays: general recommender and sequential recommender. General recommender aims to learn users' long-term preference which are presumed to be statistic or change slowly over time. Factorization-based collaborative filtering [Koren *et al.*, 2009; Koren, 2008] methods are the best-known techniques in this area. However, the intent of user behaviors is inherently variable, which can be influenced by various factors such as interest evolution, instant demands and global mainstream fashion during a certain time period. Recently, the sequential recommender has attracted considerable attention due to its superiority in capturing item-to-item sequential relations, and this superiority meets the needs of exploring users' short-term preference. Motivated by the successful applications of recurrent neural networks (RNN) in other domains, especially in Natural Language Processing (NLP), RNN approaches have now become the mainstream models for sequential recommender [Wu *et al.*, 2017; Okura *et al.*, 2017; Beutel *et al.*, 2018; Zhou *et al.*, 2019].

The vanilla RNN presumes an evenly distribution pattern between successive items in a sequence. For example, in NLP applications intervals between any two successive words in a sentence can be regarded as equal, and the whole sentence is organized for expressing the same (semantic) topic. However, a user's behavior sequence in the real-world is much more complex, and in this paper, we focus on two key problems: dynamic time intervals and dynamic latent intent. The first problem indicates the time interval between a user's successive actions can be varying. e.g., his/her next action may happen on the next day or in the next week. Intuitively, two actions within a short time interval tend to share a closer relationship than two actions that within a long time interval. Thus this kind of temporal distance deserve special handling. On the other side, the dynamic latent intent problem indicates a type of semantic distance. Customer intent, also known as the user's main purpose behind his/her behavior, is often changing from session to session. Irrelevant actions are useless for predicting a user's certain future action. For example, suppose a user's purchase history is [*iPhone xs*, *airpods*, *yogurt X*, *cat food X*], when we want to recommend him with laptops, only the first two actions, i.e. the *iPhone xs* and *airpods*, are strongly signals to infer that he/she may be interested in MacBook series. Some recent works partially discuss these two challenges [Zhu *et al.*, 2017; Zhou *et al.*, 2019], but none of them address both dynamic time interval and latent intent problems in a unified model.

Motivated by the aforementioned observations, we first propose a novel sequential recommender based on LSTM [Hochreiter and Schmidhuber, 1997], named *TC-SeqRec*, for better short-term user modeling. There are two key mecha-

*Corresponding authors.

nisms in our TC-SeqRec model, corresponding to time-aware controller and content-aware controller. Specifically, to cope with the time-aware distance, we propose to utilize the gated mechanism in classical LSTM, and it is capable of controlling to what extent the information should be translated to future stages according to time context. To cope with the content-aware distance, we adapt attention mechanisms to dynamically filter out irrelevant actions in a user's behavior sequence, since attention mechanisms have achieved remarkable results in multiple domains such as recommender systems, question answering, machine translation, and image captioning. Via jointly training with both the time-aware and content-aware mechanisms, TC-SeqRec is more effective and robust for short-term user modeling.

Usually we do not merely rely on RNNs for sequential recommender systems. Although users' interest may be time-varying, there is still some static components influencing users' behaviors, which reflect their long-term preferences. That is why existing approaches tend to combine both short-term and long-term interaction [Rendle *et al.*, 2010; Wu *et al.*, 2017]. We find that a naive and static combination of these two aspects are insufficient. The aforementioned time-aware and content-aware observations can also be applied in this combination stage. We propose an attention-based fusion method to adaptively incorporate the short- and long-term preferences under specific circumstances, such as when (if the next action occurs shortly after the last action, short-term information may play a major role in prediction; otherwise, we should shift more weight to the long-term component) and what (if recent actions share a distinct intent, then the next action may have a higher probability to share the same intent). We name the integrated model ***SLi-Rec***, which is short for Short-term and Long-term preference Integrated RECommender system. We conduct extensive experiments on both a public dataset and our industrial dataset, and results demonstrate that our proposed model outperforms several state-of-art models significantly.

2 Related Work

2.1 General Recommender

A general recommender models users' long-term preference which reflects users' inherent characteristics and are static or changed slowly with time. Items recommended by this type of recommender are supposed to be interacted with by the user eventually without any consideration of the order. Factorization-based collaborative filtering methods [Koren *et al.*, 2009; Koren, 2008] are one of the most effective techniques for this goal. Some popular topics under this direction include unifying collaborative and content-based filtering [Basilico and Hofmann, 2004], implicit feedback [Rendle *et al.*, 2009], general user profiling [Zhong *et al.*, 2015] and so on. In recent years, deep learning techniques have been widely and successfully applied in RS, for example, [He *et al.*, 2017] proposes NCF to learn an arbitrary function from data by replacing the inner product with a neural architecture, [Cheng *et al.*, 2016] jointly trains wide linear models and deep neural networks for combining memorization and generalization, [Lian *et al.*, 2018] aims to learn effective higher-order feature interactions based on factorization machine.

2.2 Sequential Recommender

In contrast, a sequential recommender pays attention to the order of user behaviors. Some early work considers Markov chains to model sequential behavior by learning a transition graph over times that is used to predict the next action based on the recent actions of a user [Rendle *et al.*, 2010]. Recently, recurrent neural networks (RNN) have demonstrated great power in sequential recommender, including in learning user/item evolution [Wu *et al.*, 2017], building session-based RS [Hidasi *et al.*, 2015; Wang *et al.*, 2018] and next item/basket prediction [Li *et al.*, 2017; Zhu *et al.*, 2017]. [Liu *et al.*, 2016] employs adaptive context-specific input matrices and transition matrices to improve conventional RNN models. [Zhu *et al.*, 2017] proposes several time gates to model time intervals with the goal of capturing both of users' long-term and short-term interests. The sequential recommender is not only popular in academia, but also popular in industrial systems [Okura *et al.*, 2017; Beutel *et al.*, 2018; Zhou *et al.*, 2019]. In this paper, we leverage RNN structures for model users' short-term preference, and further propose an attention-based adaptive fusion schema to dynamically combine users' both short-term and long-term preference.

3 Our Model

Let $U = \{u_1, u_2, ..., u_n\}$ denote a set of users and $I = \{i_1, i_2, ..., i_m\}$ denote a set of items. A user's behavior history is represented by an ordered list: $B(u) = \{(i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|B(u)|}^u, t_{|B(u)|}^u)\}$, where (i_1^u, t_1^u) means a user u interacted with an item i_1^u at time t_1^u, $t_i < t_j$ for $i < j$, and $|B(u)|$ denotes the number of actions in the user's behavior sequence. Our task belongs to the category of embedding-based user modeling, i.e., given $B(u)$ we will generate a dense user vector $\mathbf{p}_u = \langle p_1, p_2, ..., p_d \rangle$ which encodes the user's preference and can be used (together with other rich features) to predict which item he/she will interact at a future time t_p, where $p = |B(u)| + 1$. To make user vectors be more expressive, we will introduce a method to incorporate both long-term and short-term preference. In real industrial recommender systems, users' long-term preference can be learned based on a relatively longer time range and computed offline. On the other side, users' short-term preference can be inferred at online serving based on a relatively shorter range of recent behaviors to reduce computational cost. Next, we will introduce our proposed model in detail in three parts: short-term modeling, long-term modeling, and adaptive fusion stage.

3.1 Short-Term Modeling

Due to their remarkable ability in sequential user modeling, RNNs have been attracting great attention recently in both academia [Zhu *et al.*, 2017; Wu *et al.*, 2017; Donkers *et al.*, 2017] and industry [Beutel *et al.*, 2018; Okura *et al.*, 2017; Zhou *et al.*, 2019]. The sequential updating process is very simple and can be formulated by:

$$h_k = g(x_k W + h_{k-1} U + b) \tag{1}$$

Where g is the activation function, x_k is the latest user action and h_{k-1} is the last hidden state. Among all RNN-based models, LSTM (long short-term memory) [Hochreiter and Schmidhuber, 1997] and GRU (gated recurrent unit) [Cho *et al.*, 2014] are most commonly used for RS. On our own industrial dataset we observed that LSTM is slightly better than GRU, without loss of generality, in this paper we formulate the model with LSTM, whose equations are as follows:

$$f_k = \sigma(x_k W_f + h_{k-1} U_f + b_f) \tag{2}$$
$$i_k = \sigma(x_k W_i + h_{k-1} U_i + b_i) \tag{3}$$
$$c_k = f_k \odot c_{k-1} + i_k \odot \phi(x_k W_c + h_{k-1} U_c + b_c) \tag{4}$$
$$o_k = \sigma(x_k W_o + h_{k-1} U_o + b_o) \tag{5}$$
$$h_k = o_k \odot \phi(c_k) \tag{6}$$

where $W_*, U_* \in \mathbb{R}^{D \times D}$ are trainable parameters, D indicates the dimension of input embedding and hidden layer in RNN (for notation simplicity, we presume the size of dimensions are equal). f_k, i_k, o_k represent the forget, input, and output gates, respectively. c_k represents the cell status, x_k denotes the k-th item's input embedding, and $\odot$ denotes the element-wise product. Usually, σ is the $sigmoid$ function, and ϕ is the $tanh$ function.

Different from words of a sentence in NLP domain, where items can be regarded as evenly spaced and semantically consistent, the sequence of user behaviors are much more complex. Here we mainly focus on these two challenges: (1) Time irregularity. Time intervals between two successive actions can be various. E.g., user a's purchase history is $B^a = \{(i_1^a, Feb\ 1^{st}), (i_2^a, Feb\ 2^{nd}), (i_3^a, Apr\ 2^{nd})\}$, it's more reasonable to transmit more information from i_1^a to i_2^a than from i_2^a to i_3^a, because item i_2^a is purchased just one day after purchasing item i_1^a, while item i_3^a is purchased two months after last purchase. (2) Semantic irregularity. Items within a user's behavior sequence may not always share the same semantic topic (which we usually call $customer\ intent$). E.g., user a's purchase sequence may contains items $\{iphonexs, airpods, lawnmower, Bikini\}$, these items represent his/her different demands and only the first two items are strong signals for predicting his/her next activity for electronic products.

To tackle the problem of time irregularity, we modify the gating logic in LSTM to make it sensitive to time changes. We introduce two time-aware features, i.e., time interval feature δ_{t_k} and time span feature s_{t_k}, as follows:

$$\delta_{t_k} = \phi(W_\delta log(t_k - t_{k-1}) + b_\delta) \tag{7}$$
$$s_{t_k} = \phi(W_s log(t_p - t_k) + b_s) \tag{8}$$
$$T_\delta = \sigma(x_k W_{x\delta} + \delta_{t_k} W_{t\delta} + b_{t\delta}) \tag{9}$$
$$T_s = \sigma(x_k W_{xs} + s_{t_k} W_{ts} + b_{ts}) \tag{10}$$

where W_δ, $W_s \in \mathbb{R}^D$ and $W_{t\delta}$, $W_{ts} \in \mathbb{R}^{D \times D}$. The time span feature s_{t_k} encodes the absolute temporal distance between current state t_k and prediction state t_p, while the time interval feature δ_{t_k} encodes the relative temporal distance between two consecutive states. Similar to [Beutel *et al.*, 2018], we add a fully connected layer to convert the time-aware features into dense vectors ($\delta_{t_k} W_{t\delta}$ and $s_{t_k} W_{ts}$), then compute time gates (T_δ and T_s) accordingly. Eq.(4) is now changed to:

$$\begin{aligned} c_k =& f_k \odot T_\delta \odot c_{k-1} \\ &+ i_k \odot T_s \odot \phi(x_k W_c + h_{k-1} U_c + b_c) \end{aligned} \tag{11}$$

and Eq.(5) is updated as:

$$o_k = \sigma(x_k W_{xo} + \delta_{t_k} W_{\delta o} + s_{t_k} W_{so} + h_{k-1} W_{ho} + b_o) \tag{12}$$

To tackle the problem of semantic irregularity, we adopt attentive mechanisms to suppress the information that deviates from the target direction. Actually, attention mechanisms have been widely used to filter out irrelevant items or distinguish different levels of influence scores for relevant items [Wang *et al.*, 2018; Ying *et al.*, 2018; Zhou *et al.*, 2019; Li *et al.*, 2017]. An item's attention score is computed by:

$$a_k = \frac{exp(x_k W_x^s e_p)}{\sum_{j=1}^{|B_u|} exp(x_j W_x^s e_p)} \tag{13}$$

where e_p represents another embedding vector of item we want to make prediction, we call it *prediction embedding* in contrast to the *input embedding* vector x_p. The attention score a_k determines which item should be emphasized or neglected according to target items. We utilize these scores to adjust the cell and hidden states:

$$\widetilde{c_k} = a_k * c_k + (1 - a_k) * c_{k-1} \tag{14}$$
$$\widetilde{h_k} = a_k * h_k + (1 - a_k) * h_{k-1} \tag{15}$$

Instead of using the last hidden state as user representation, i.e., $p_{short}^u = \widetilde{h_k}$, we formulate user's short term representation as the weighted average of all the hidden states:

$$a_k^s = \frac{exp(\widetilde{h_k} W_h^s e_p)}{\sum_{j=1}^{|B_u|} exp(\widetilde{h_j} W_h^s e_p)} \tag{16}$$
$$p_u^{short} = \sum_{j=1}^{|B_u|} a_j^s \widetilde{h_j} \tag{17}$$

We call our proposed short-term model ***TC-SeqRec***, which indicates that the model is a time- and content-aware sequential recommender.

3.2 Long-Term Modeling

In this component we aim to model users' general preference that are inherent and supposed to be static or changed slowly. Matrix factorization techniques are the most successful methods for learning this long-term preference. Instead of providing an explicit parameterization for users, we adopt the attentive "*Asymmetric-SVD*" [Koren, 2008] paradigm, which represents users through the items that they interacted with:

$$p_u^{long} = \sum_{j \in B(u)} a_j^l x_j \tag{18}$$

similar to the Attentive Collaborative Filtering [Chen *et al.*, 2017], a_j^l is the weighting score for behavior j. We presume that not all behaviors contribute equally, thus it is meaningful

to assign higher (lower) weights for the corresponding behaviors that are more informative (less informative). The weighting score is computed as:

$$v_k = \phi(W_v^l x_k + b_v) \tag{19}$$

$$a_k^l = \frac{exp(v_k \tau_l)}{\sum_{j \in B(u)} exp(v_j \tau_l)} \tag{20}$$

where $W_v^l \in \mathbb{R}^{D \times D}$ and $\tau_l \in \mathbb{R}^D$. Usually we can add a general user vector to the Asymmetric-SVD, i.e., $p_u^{long} = p'_u + \sum_{j \in B(u)} a_j^l x_j$. But to reduce parameters and prevent overfitting, we merely use Eq.(18) as user representation.

3.3 Adaptive Fusion Approach

Both short-term and long-term components have strengths and weaknesses. It is necessary to accommodate these two components. Instead of using a naive way to combine them, e.g., $p_u^{final} = p_u^{short} + p_u^{long}$, we design an adaptive way for information fusion. The motivation is which component should play a more important role is determined by the specific context, such as *when* (if next action is taken shortly after the previous behaviors, then short-term preference may be more informative) and *what* (some categories of items such as mobile phone are better inferred from long-term preference, while some categories such as mobile accessories are better inferred from short-term information). Thus information fusion in a dynamic fashion is beneficial, and we propose the following attention-based adaptive fusion method:

$$\alpha = \sigma(W^m [p_u^{short}, p_u^{long}, x_{context}] + b_m) \tag{21}$$

$$p_u^{final} = \alpha * p_u^{short} + (1 - \alpha) * p_u^{long} \tag{22}$$

where $[p_u^{short}, p_u^{long}, x_{context}]$ represents a concatenation of short-term information, long-term information, and the contextual information. Note that for contextual information, we can include various kinds of important features according to what we have at hand, such as time interval, time stamp, location, and target item category. In our experiments we only include the item interval and the prediction embedding of the target item. We name the model ***SLi-Rec***, which is short for Short-term and Long-term preference Integrated RECommender system.

A typical function for measuring user and item interaction is the doc product in the form of $\hat{y}_{ui} = \langle p_u^{final}, e_i \rangle$. A more flexible approach is to feed them into a multilayer perception (MLP). In this way, various additional features, such as user profiles and contextual features, can be easily incorporated for industrial recommender systems. To emphasize the impact of different user representations, in this paper we only concatenate the user vector and item vector as the input for a two-layer MLP, and all compared models in the experiment section will share this design, i.e., $\hat{y}_{ui} = MLP([p_u^{final}, e_i])$.

Because our real-world industrial task is related to CTR (click-through rate) prediction, we formulate the recommendation task as a binary classification problem, where the negative log-likelihood function (log-loss) is usually used as the loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} y_i log \hat{y}_i + (1 - y_i) log(1 - \hat{y}_i) \tag{23}$$

Dataset	Users	Items	Category	Instances
Electronics	192k	63k	704	2993k
Movies	123k	50k	163	3147k
CDs	75k	64k	343	2044k
Entire	100k	1,200k	12024	6716k
Ads	519k	668k	-	3894k

Table 1: Basic statistics of the datasets. Category feature is not enable in *Ads* dataset. "k" indicates a thousand.

where N is the total number of training instances. $y_i = 1$ indicates a positive instance (the user has interacted with the item) and $y_i = 0$ indicates a negative instance. The optimization process is to minimize the loss function together with a regularization term:

$$\mathcal{J} = \mathcal{L} + \lambda_* ||\Theta||_2 \tag{24}$$

where Θ denotes the set of trainable parameters.

4 Experiments

4.1 Datasets

We use both a public dataset and our own industrial dataset for experimental analysis. The basic statistics of datasets (after filtering) are shown in Table 1.

Amazon dataset. [He and McAuley, 2016]. This is a public dataset containing product reviews and metadata from Amazon. Nowadays it's widely used as a benchmark dataset in the RS domain. Reviews can reveal customers' shopping behaviors. We choose three subcategories, i.e., Electronics, Movies and TV, CDs and Vinyl, as well as a subset of the entire merged Amazon dataset for experiments. Item is represented by item id and category id. Given a user's previous T behaviors, we want to predict his T+1 behavior. For each user, his/her last behavior is hold out as the test set. For each positive instance we randomly sample one negative instance. 50% of the test set is used for hyperparameter tuning and the rest is used for reporting evaluation metrics.

Industrial dataset. Our real application scenario is native advertising [1]. We display personalized advertisements on MSN[2] homepage in a fashion that advertisements look like part of the editorial flow of the page. A user's behavior sequence is comprised of an ordered list of general browsing records. Thus the input embedding (i.e., x_k) is derived from titles of browsing records while prediction embedding (i.e., e_k) is from advertisement items. We extract the impression logs from 10 consecutive days in November 2018, and downsample the non-click logs to make the ratio of positive instances to negative instances be 1:5. The first 7 days are used as the training set, and the latter 3 days are used for the test set. Similar to the *Amazon* settings, 50% of the test set is used for hyperparameter tuning and the rest is used for reporting evaluation metrics.

[1] https://en.wikipedia.org/wiki/Native_advertising

[2] https://www.msn.com/

4.2 Compared Methods

We compare SLi-Rec with the following competitive models:
ASVD [Koren, 2008] represents users by the items that they have interacted with. Items contribute equally. We further use Eq.(18) to get an attentive ASVD, or **A^{2}SVD** for short.
DIN [Zhou *et al.*, 2018] uses an attentive mechanism to dynamically activate related items in a user's behavior history according to the target item.
LSTM The classical LSTM model for sequential prediction.
NARM [Li *et al.*, 2017] is a neural attentive recommendation machine, which captures a user's main purpose in the current session by incorporating an attention mechanism into RNN.
RRN [Wu *et al.*, 2017] also leverages LSTM to capture dynamics in addition to a traditional low-rank factorization. The biggest difference between this method and the *LSTM* baseline is that, in RRN the user-item interactions are aggregated by a time step granularity. We have tried the granularity of daily and weekly, denoted by *RRN-day* and *RRN-week*.
LSTM++ combines the *A^{2}SVD* and the *LSTM* baselines to capture both long-term and short-term preferences. We denote it as *LSTM++* for convenience.
CA-RNN [Liu *et al.*, 2016] is the context-aware recurrent neural networks which employs adaptive context-specific input matrices and transition matrices in the RNN framework.
T-LSTM [Zhu *et al.*, 2017] equips LSTM with time gates to model time intervals. Different from our model, it doesn't handle time span and relies on the time gates to capture both long-term and short-term interests.
DIEN [Zhou *et al.*, 2019] is the most recent related work which models the user's sequential behaviors with two layers of GRU. Its key components consist of interest extracting layer and interest evolving layer. The long-term component in this baseline is ASVD.
We implement the models with Tensorflow, the source code is available at https://github.com/zepingyu0512/sli_rec. Optimizer is Adam. Dimension for item/category embedding and RNN hidden layers is 18, while the dimension for MLP is 36. We adopt grid search to find the best hyperparameters for each model with validation set. Optimal settings for our model are: *learning rate* is 0.001; L_2 regularizataion is 0.0001; no *dropouts*; *batch normalization* is used only after the concatenation of the user's (final) embedding and item embedding; activation function of MLP is Dice [Zhou *et al.*, 2018] for Amazon and ReLU for Ads dataset; maximum length for user behaviors is set to 100.

Evaluation metrics. In Section 3.3 we formulate the recommendation task as a binary classification problem, thus we use AUC and F1-score as the evaluation metrics. AUC measures the probability that a positive instance will be ranked higher than a randomly chosen negative one. F1-score is the harmonic mean of precision and recall. These metrics summarize a model's performance from different aspects.

4.3 Results

Comparison to baselines. Tables 2 and 3 show the overall performance of different models in terms of AUC and F1-score respectively. *ASVD*, *A^{2}SVD* and *DIN* are models without sequential mechanism, while the other models have a sequential component. *A^{2}SVD* and *DIN* outperform *ASVD*, which demonstrates that assigning importance score to items are beneficial. Containing both short-term and long-term interest is meaningful for recommender systems, and this can easily be verified via comparing *LSTM++* with *LSTM*. *NARM* is original designed for capturing the user's main purpose in the current session, which is different from our application scenario, thus its performance is not very outstanding in our cases. *CARNN*, *T-LSTM*, and *DIEN* can outperform *LSTM* and *RRN* in most cases, which directly verify that considering contextual information in sequence is helpful. At last, our proposed *SLi-Rec* significantly outperform all the baselines on five datasets across the three evaluation metrics. The superiority of *SLi-Rec* depends on three components, i.e., the short-term part, the long-term part, and the adaptive fusing part. Next we will investigate the effectiveness of these components separately.

Model	Entire	Elec	Movies	CDs	Ads
ASVD	0.8060	0.7727	0.8156	0.8863	0.6463
A^{2}SVD	0.8204	0.7838	0.8263	0.9032	0.6501
DIN	0.8293	0.7927	0.8388	0.9111	0.6520
LSTM	0.8272	0.7859	0.8414	0.9086	0.6527
LSTM++	0.8306	0.8028	0.8495	0.9113	0.6535
NARM	0.8290	0.7876	0.8448	0.9130	0.6531
RRN$_{day}$	0.8260	0.7864	0.8406	0.9078	0.6531
RRN$_{week}$	0.8250	0.7869	0.8390	0.9069	0.6500
CARNN	0.8278	0.8106	0.8527	0.9096	0.6551
T-LSTM	0.8387	0.8212	0.8660	0.9181	0.6597
DIEN	0.8361	0.7904	0.8438	0.9128	0.6610
SLi-Rec	**0.8494**	**0.8282**	**0.8769**	**0.9279**	**0.6654**

Table 2: Performance comparison in terms of AUC. A **bold** font means the number is significantly bigger than the second best model with p-value < 0.05. For notation simplicity we omit the asterisks and the same goes for all the other tables.

Model variants for short-term component. To verify whether our design of time-aware and content-aware con-

Model	Entire	Elec	Movies	CDs	Ads
ASVD	0.7427	0.7255	0.7539	0.8128	0.3242
A^{2}SVD	0.7538	0.7264	0.7565	0.8264	0.3270
DIN	0.7599	0.7349	0.7660	0.8348	0.3320
LSTM	0.7556	0.7311	0.7683	0.8325	0.3296
LSTM++	0.7591	0.7448	0.7742	0.8352	0.3338
NARM	0.7565	0.7323	0.7706	0.8375	0.3327
RRN$_{day}$	0.7550	0.7318	0.7685	0.8322	0.3313
RRN$_{week}$	0.7545	0.7327	0.7676	0.8315	0.3313
CARNN	0.7584	0.7519	0.7763	0.8336	0.3292
T-LSTM	0.7591	0.7448	0.7742	0.8352	0.3338
DIEN	0.7632	0.7327	0.7755	0.8374	0.3343
SLi-Rec	**0.7745**	**0.7643**	**0.7958**	**0.8532**	**0.3367**

Table 3: Performance comparison in terms of F1-score.

Model	Entire	Elec	Movies	CDs	Ads
LSTM	0.8272	0.7859	0.8414	0.9086	0.6527
T-LSTM	0.8387	0.8212	0.8660	0.9181	0.6597
T-SeqRec	0.8401	0.8248	0.8709	0.9223	0.6615
TC-SeqRec$_i$	0.8324	0.8178	0.8602	0.9080	0.6596
TC-SeqRec$_g$	0.8356	**0.8261**	0.8696	0.9220	**0.6634**
TC-SeqRec	**0.8453**	**0.8264**	**0.8730**	**0.9244**	**0.6639**

Table 4: Comparison of variants of short-term models (in AUC).

α design	Entire	Elec	Movies	CDs	Ads
0	0.8204	0.7838	0.8263	0.9032	0.6501
1	0.8453	0.8264	0.8730	0.9244	0.6634
fixed	0.8397	**0.8274**	0.8725	0.9261	0.6641
adaptive	**0.8494**	**0.8282**	**0.8769**	**0.9279**	**0.6654**

Table 5: Comparison of different fusing methods (in AUC).

trollers are necessary and effective for short-term preference modeling, we study several variants. Table 4 shows the results. All the other models are better than *LSTM*, which demonstrates that consider complex user behavior patterns is indeed meaningful. *T-SeqRec* is our short-term model which only enable the time-aware controller. Compared with *T-LSTM*, *T-SeqRec* not only uses time interval, but also uses time span feature. *TC-SeqRec*$_*$ are different approaches to enable the content-aware controller, where i means the item attention score (Eq.(13)) is applied to input features instead of to cell and hidden states, g means we use the last state as short-term preference instead of averaging all the hidden states (Eq.(17)). Via comparing with these various variants, we can observe that both the time-aware and content-aware controllers are beneficial to short-term user modeling.

Effectiveness of adaptive fusion. Table 5 demonstrates various choices of α in Eq.(22). $\alpha = 0$ and $\alpha = 1$ indicate only using a long-term or short-term component, respectively. $\alpha = fixed$ means we empirically search a fixed optimal value for each dataset (similar to an end-2-end ensemble of two components). We observe that the short-term component is always significantly better than the long-term component, thus sometimes a naive combination of these two components turns out to be even worse than a mere short-term component. An adaptive fusion mechanism is necessary and effective. To get an intuitive sense of how α would change with context, we split test instances according to the time interval of a user's last behavior. For instance, *1 hour* means than the behavior which we want make prediction occurs within one hour after the user's last behavior. Figure 1 shows the results. There is a clear trend that importance for long-term component increases with the time interval, which matches our assumption.

Attention in long-term component. To verify whether Eq.(18) can learn discriminative importance scores for different items, we output item's weighting values in Eq.(20)

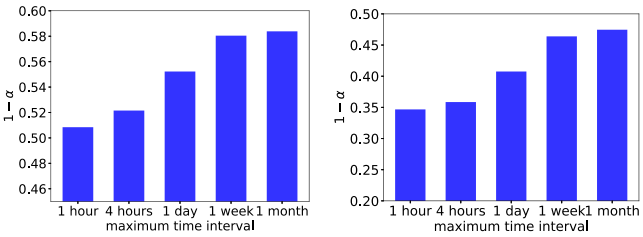
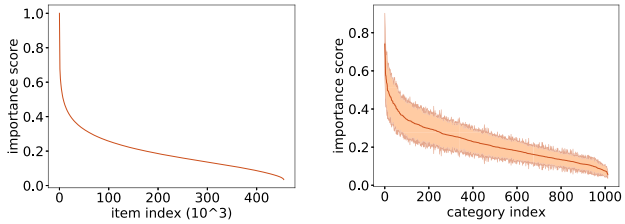


Figure 1: Average integration values (Eq.(21)) for Long-term preference component on the CDs (left) and Movies (right) dataset $w.r.t.$ the next prediction time interval.



(a) Item (b) Category (with variance).

Figure 2: Distribution of attentive weights (Eq.(20)) for items and categories (item average) in long-term preference modeling.

on the *entire* dataset. For better illustration, for each category we at most sample 1000 items, and ignore categories that contain fewer than 100 items. We sort items by their normalized importance score and plot Figure 2a. For each category, we count the mean and variance of all items' scores under this category and draw Figure 2b (the scores are scaled by *1e-5*). We can observe that items are assigned with discriminative scores, and few items (less than 2%) are assign a normalized score greater than 0.5.

5 Conclusion

In this paper, we have proposed the novel SLi-Rec model for integrating both the short-term and long-term preference for better user modeling. We observe that users' behavior sequences are much more complex and challenging than sequences in other application domains (such as sentences in NLP), thus we propose the time-aware and content-aware controllers to make the classical LSTM more suitable for user behavior modeling. We further propose an attention-based fusion method to adaptively combine the long-term and short-term preference according to the specific context. We have conduct extensive experiments on both public dataset and industrial dataset, experiments demonstrate that our proposed model outperforms state-of-the-art methods consistently.

Acknowledgments

We gratefully thank the Microsoft Audience Network (MSAN) team, especially Yajun Wang and Mehul Parsana for their contributions and support for experiments and applications in native advertisement.

References

[Basilico and Hofmann, 2004] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9. ACM, 2004.

[Beutel *et al.*, 2018] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54. ACM, 2018.

[Chen *et al.*, 2017] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344. ACM, 2017.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Donkers *et al.*, 2017] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160. ACM, 2017.

[He and McAuley, 2016] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

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

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

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

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

[Lian *et al.*, 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1754–1763. ACM, 2018.

[Liu *et al.*, 2016] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1053–1058. IEEE, 2016.

[Okura *et al.*, 2017] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942. ACM, 2017.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

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

[Wang *et al.*, 2018] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Wu *et al.*, 2017] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503. ACM, 2017.

[Ying *et al.*, 2018] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention networks. In *the 27th International Joint Conference on Artificial Intelligence*, 2018.

[Zhong *et al.*, 2015] Erheng Zhong, Nathan Liu, Yue Shi, and Suju Rajan. Building discriminative user profiles for large-scale content recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2277–2286. ACM, 2015.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1059–1068, New York, NY, USA, 2018. ACM.

[Zhou *et al.*, 2019] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[Zhu *et al.*, 2017] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 3602–3608. AAAI Press, 2017.