CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation

Ting Bai[1,2*], Lixin Zou[3*], Wayne Xin Zhao[1,2,†], Pan Du[4]
Weidong Liu[3], Jian-Yun Nie[4], Ji-Rong Wen[1,2]
[1]School of Information, Renmin University of China, Beijing, China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
3 Department of Computer Science and Technology, Tsinghua University, Beijing, China
[4] Department of Computer Science and Operations Research, University of Montreal, Canada
baiting@ruc.edu.cn,{zoulixin15,batmanfly,jirong.wen}@gmail.com
liuwd@tsinghua.edu.cn,{nie,pandu}@iro.umontreal.ca

ABSTRACT

In e-commerce, users' demands are not only conditioned by their profile and preferences, but also by their recent purchases that may generate new demands, as well as periodical demands that depend on purchases made some time ago. We call them respectively short-term demands and long-term demands. In this paper, we propose a novel self-attentive Continuous-Time Recommendation model (CTRec) for capturing the evolving demands of users over time. For modeling such time-sensitive demands, a Demand-aware Hawkes Process (DHP) framework is designed in CTRec to learn from the discrete purchase records of users. More specifically, a convolutional neural network is utilized to capture the short-term demands; and a self-attention mechanism is employed to capture the periodical purchase cycles of long-term demands. All types of demands are fused in DHP to make final continuous-time recommendations. We conduct extensive experiments on four real-world commercial datasets to demonstrate that CTRec is effective for general sequential recommendation problems, including next-item and next-session/basket recommendations. We observe in particular that CTRec is capable of learning the purchase cycles of products and estimating the purchase time of a product given a user.

CCS CONCEPTS

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

KEYWORDS

Continuous-Time Recommendation; Long-Short Demands; Demand-Aware Hawkes Process; Self-Attentive Mechanism

* Both authors contributed equally to this research
† Corresponding author.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331199

ACM Reference Format:
Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, Ji-Rong Wen. 2019. CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation. In *SIGIR'19: The 42nd International ACM SIGIR Conference on Research Development in Information Retrieval, July 21-25, 2019, Paris, France.* ACM, NY, NY, USA, 10 pages.
https://doi.org/10.1145/3331184.3331199

1 INTRODUCTION

Recommender systems provide great help for users to find their desired items from a huge number of offers. So far, the majority of recommendation models, *e.g.,* collaborative filtering [3, 12, 16] and sequence-based models [20, 26, 37], have mainly focused on modeling user's general interests to find the right products, while the aspect of meeting users' purchase demands at the right time has been much less explored. We believe that a good recommender system should not only be able to find the right products, but also recommend them at the right time to meet the demands of users, so as to maximize their values and enhance the user experiences [9]. Otherwise, one could expect complaints from customers, as a recent one on Amazon[1] for always sending her promotion emails on toilet seat after she bought one[2] : *"Dear Amazon, I bought a toilet seat because I needed one. Necessity, not desire. I do not collect them. I am not a toilet seat addict"*. This problem occurred because the recommender system inferred that the user is generally interested in toilet seats due to her recent purchase. But it ignored the fact that the user's demand is satisfied when she had bought one, and the same or similar product should be recommended again only after the service life of the old one expiring. Without considering purchase demands, it is difficult to determine the right time for a recommendation, which may lead to the above toilet-seat situation. Purchase demands of users are highly time-sensitive: it may be affected by recent purchases and purchases made some time ago. In addition, other factors such as item attributes are likely to affect the purchase demands. All these elements should be taken into account together with time. To the best of our knowledge, most of the previous studies mainly focus on the learning of user interests, and they seldom consider satisfying user demands at proper time in the final purchase decision.

[1]https://www.amazon.com/
[2]https://twitter.com/GirlFromBlupo/status/982156453396996096

In this paper, we address this problem by proposing a Continuous-Time Recommendation model (CTRec) to capture the evolving users' purchase demands over time. Our model is designed to capture the complex time-sensitive correlations among items, and to leverage such information to better detect the current demand of a user, so as to make a more accurate prediction at a certain future time. Inspired by the studies in marketing strategies and human behaviors [25, 27, 33], we consider two kinds of the time-sensitive purchase demands: termed as *long-term demands* and *short-term demands*. Long-term demands refer to the persistent demands of a user of the same or similar products [4], *e.g.,* laundry detergent, which should be recommended regularly to the user according to certain service time cycle. While short-term demands of products are strongly related to the products a user purchased recently and can be observed in a relatively short time in purchase records [8], *e.g.,* buying paintbrushes after buying pigments. For modeling such long- and short-term purchase demands of users, we design a Demand-aware Hawkes Process (DHP) framework in CTRec to capture the sequential information in continuous time from the discrete purchase records of users. More specifically, we utilize a convolutional neural network component to learn the information of associated items in a short time, and employ a self-attentive component to capture the long-term purchase cycles of products. The demand-aware Hawkes process acts as follows: Once a user has purchased a product, the demand of that product is initialized to a small value, and the demand for it (or for similar product) increases slowly along time at a pace depending on the product.

Our CTRec model tries to learn the evolution of user's demands over time and estimates the probability of the demand on items at a given time for the user. We found limited research about continuous-time recommendation and none of that dealt with user's demands in real-world commercial datasets. Our contributions are as follows:

- We propose a novel continuous-time model CTRec for sequential recommendation tasks by taking time into account. A Demand-aware Hawkes Process (DHP) framework is designed for modeling demand evolution in continuous time. To the best of our knowledge, such demand-aware continuous-time model has not been explored in real-world commercial recommendation scenarios.
- We characterize two kinds of user purchase demands: long-term demands (*e.g.,* repeated purchasing with a persistent interest) and short-term demands (*e.g.,* buying the complementary purchasing in a short time period). A convolutional recurrent neural network and self-attentive components are integrated in DHP framework to capture the short- and long demands respectively.
- Extensive experiments on four real-world commercial datasets demonstrate the effectiveness of our continuous-time model for sequential recommendation, *i.e.,* next-item, next-session or basket[3] tasks, showing the usefulness of modeling users' long-short demands evolution on continuous-time. In particular, our model is capable of learning the purchase cycles of products and to estimating the real purchase time of a product given a user.

[3]We consider next-session and next-basket recommendation as the same sequential recommendation task due to the fact that both recommend a set of items.

2 PROBLEM DEFINITION

Assume we have a set of users and items, denoted by U and I respectively. For a user u, his purchase record is a sequence of items sorted by time, which can be represented as $I_{t_n} = \{i_{t_1}, i_{t_2}, ..., i_{t_j}, ..., i_{t_n}\}$, where $i_{t_j} \in I_{t_n}$ is the item purchased by user u at time t_j. Each item $i \in I$ has some attributes, *e.g.,* category, denoted as $a \in A$, where A is the set of attributes. We define the continuous-time recommendation problem as follows: given the purchase history I_{t_n} of a user u, we want to infer the probability $Pr(i_{t_{n+\epsilon}})$ of items being purchased by user u at a future time $t_{n+\epsilon}$:

$$Pr(i_{t_{n+\epsilon}}) = \mathcal{F}(i \in I | I^u_{t_n}, t_\epsilon), \tag{1}$$

where t_ϵ is the time interval from t_n to $t_{n+\epsilon}$, and $\mathcal{F}$ is the prediction function.

By considering both the order and time interval information in sequence, we formulate our continuous-time recommendation problem, *i.e.,* $\{i_{t_1}, i_{t_2}, ..., i_{t_j}, ..., i_{t_n}\} \rightarrow i_{t_{n+\epsilon}}$, as a generalized sequential recommendation problem. Both the next-item and next-session/basket problems, which only consider the ordering relation of items, can be regarded as specialized cases of continuous-time recommendation by discretizing time information. The details are discussed in Sec. 3.5.

3 THE PROPOSED MODEL

We design a Demand-aware Hawkes Process (DHP) framework in CTRec to capture the sequential information from the discrete purchase records of users. The architecture of our CTRec is shown in Fig. 1. A convolutional neural network component is utilized to capture the information of associated items for the short-term demands and self-attentive cycles component is employed for modeling the long-term demands.

3.1 General Framework

User's demands are influenced by the items they have already bought and the influence evolves as time goes. The influence in event streams have been studied in [23]: a self-modulating Hawkes process is utilized to explore the excites or inhabits from previous events in continuous time. Inspired by [23], we propose a demand-aware Hawkes process for capturing the complex influence of previous items on the future demands. It is natural to build such a continuous-time model due to the fact that users demands are highly time sensitive: it may be generated by their recent purchases, as well as the purchases made some time ago, i.e. short- and long-term demands. An important difference between our work and the event stream model [23] is that we consider the evolution of two kinds of demands over time. The incentive effect in the Hawks process may increase with a certain cyclicality for long-term demands while decrease in the short-term demands, and such influence cannot be handled in the simple event streams model [23].

3.1.1 Neural Hawkes Process. Neural hawkes process is a neurally self-modulating multivariate point process [19]. Given a user u and the purchase history $I^u_{t_n}$, let $\lambda_i(t)$ be the intensity function of an event i at time point t, the probability of occurrence for the new event i in a small time window $[t, t + dt)$ can be determined by

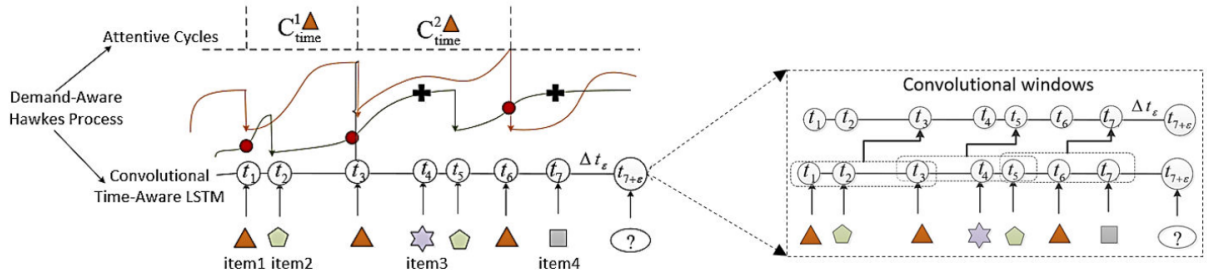

Figure 1: An illustration of our self-attentive continuous-time recommendation model (CTRec). CTRec consists of two main components: a *Convolutional Time-Aware LSTM* for capturing the *short-term* influence among items; and an *Attentive Cycles Component* for modeling the *long-term* purchase cycles of items. The influence among items over time is modeled by a Hawkes process: Red Circle means that item 1 has a positive impact on item 2, while the Black Cross refers to the negative impact of item 3 and item 4 on item 2 (*e.g.,* Buying cookies (item 1) may increase the probability of buying milk (item 2), while buying other dairy products such as yogourt (item 3) and powdered milk (item 4) may temporarily decrease the need to purchases milk); In the Attentive Cycles, a user's demand of a product,*i.e.,* item 1, follows a certain purchase cycle, *e.g.,* C^1_{time} and C^2_{time}.

$\lambda_i(t)dt$:

$$Pr(i_{t+dt}|I^u_{t_n}, dt) = \lambda_i(t)dt, \tag{2}$$

where dt is the interval of the time window.

3.1.2 Demand-Aware Hawkes Process. For modeling the time-sensitive demands of users, we design a Demand-aware Hawkes Process (DHP) in CTRec. By characterizing users' short-term demands of items as $\boldsymbol{h}(t)$ and long-term demands as $\boldsymbol{\vartheta}(t)$, the conditional intensity function $\lambda_i(t;\theta)$ can be written as

$$\lambda_i(t;\theta) = f(\underbrace{\boldsymbol{w}_i^{item\top} \cdot \boldsymbol{h}(t)}_{\text{short-term}} + \underbrace{\boldsymbol{w}_i^{attri\top} \cdot \boldsymbol{\vartheta}(t)}_{\text{long-term}} + \underbrace{\boldsymbol{w}_i^{user\top} \cdot \boldsymbol{u}}_{\text{basic demands}}), \tag{3}$$

where θ is the parameters of our model, $f : \mathbb{R} \to \mathbb{R}^+$ is the transfer function to obtain a positive intensity function, and $f(x) = \frac{s}{1+\exp(\frac{-x}{s})}$, where s is set to 5 in our experiment for the optimal results, $\boldsymbol{w}_i^{item}$, $\boldsymbol{w}_i^{attri}$ and $\boldsymbol{w}_i^{user}$ are the learned weights for different aspects for users' demands of item i.

The meaning of each module is as follows:

- $\boldsymbol{w}_i^{item\top} \cdot \boldsymbol{h}(t)$ represents the users' short-term demands of items (*i.e.,* item-level influence) from the historical events.
- $\boldsymbol{w}_i^{attri\top} \cdot \boldsymbol{\vartheta}(t)$ emphasizes the influence of long-term demands of items with some common attributes. It is very common that a user may purchase the similar items (*e.g.,* with the same category or brand) rather than exactly the same one periodically. Hence we consider the attribute-level influence of items.
- $\boldsymbol{w}_i^{user\top} \cdot \boldsymbol{u}$ refers to users' base interests of purchasing item i at any time.

Given the intensity function $\lambda_i(t;\theta)$, the probability $p_i(t;\theta)$ (*i.e.,* $Pr(i_t|I^u_{t_o}, \Delta t)$ in Eq. 1) of item i will be purchased at time t can be represented as:

$$p_i(t;\theta) = \lambda_i(t;\theta) \exp\left(-\int_{t_o}^{t} \lambda_i(s;\theta)ds\right), \tag{4}$$

where t_o is the last observed time of purchase history, and $s \in [t_o, t]$.

As demonstrated in [23], the expectation next purchase time $\hat{t}_{next}$ of item i is:

$$\hat{t}_{next} = \int_{t_o}^{+\infty} t \cdot p_i(t;\theta)dt. \tag{5}$$

In general, the integration does not have analytic solutions, we estimate the value with the Monte Carlo trick to handle the integral.

3.2 Modeling Short-Term Demands by Convolutional Time-Aware LSTM

Users' short-term demands can be regarded as the local sequential patterns among items within a close proximity of time [32], *e.g.,* a user likely buys mouse soon after buying a laptop. For better capturing the local sequential patterns, we utilize a convolutional time-aware LSTM for modeling the short-term demands of users.

3.2.1 Convolutional Representation with Local Sequential Patterns. We represent each item with a convolutional filter of items in a close time window k [32] (see in right part of Fig. 1). An item i_{t_j}, together with the previous $k-1$ items, generates a new convolutional representation $\boldsymbol{\iota}^k$. For the first $k-1$ item, we use fake-items (*i.e.,* zero vectors) for auto-completion. We utilize multiple window size, *i.e.,* $k \in \{k_1, k_2, \ldots, k_m\}$, to learn the different local features. Then we conduct average-pooling on the multi-filter convolutional representations, the final convolutional vector for item i_{t_j} is $\boldsymbol{v}_{t_j}$, defined as:

$$\boldsymbol{v}_{t_j} = avg\{\boldsymbol{\iota}^{k_1}_{t_j}, \ldots, \boldsymbol{\iota}^{k_m}_{t_j}\}. \tag{6}$$

The convolutional vectors of all items can be represented as $\boldsymbol{v} = \{\boldsymbol{v}_{t_1}, \ldots, \boldsymbol{v}_{t_n}\}$. We feed convolutional representations of items into a time-aware LSTM to capture the evolving for short-term demands.

3.2.2 Time-Aware LSTM. Traditional Recurrent Neural Networks (RNN), only consider the sequential order of objects with discrete time-steps. Inspired by [23], in our work, we utilize a time-aware LSTM to compose the intensity function of DHP framework over continuous time. In time-aware LSTM, the input is the convolutional

representations of items, *i.e.,* $\boldsymbol{v} = \{\boldsymbol{v}_{t_1}, \ldots, \boldsymbol{v}_{t_n}\}$. The hidden state vector $\boldsymbol{h}(t) \in \mathbb{R}^D$ depends on the vector $\boldsymbol{c}(t) \in \mathbb{R}^D$ of memory cells, which exponentially decays with time interval $t - t_k$ at rate $\boldsymbol{\delta}_{k+1}$ toward a steady-state value $\bar{\boldsymbol{c}}_{t_{k+1}}$ as follows:

$$\boldsymbol{c}(t) = \bar{\boldsymbol{c}}_{k+1} + \left(\boldsymbol{c}_{k+1} - \bar{\boldsymbol{c}}_{k+1}\right) \exp\left(-\boldsymbol{\delta}_{k+1}\left(t - t_k\right)\right) \quad (7)$$

$$\boldsymbol{h}(t) = \boldsymbol{o}_k \odot \left(2\sigma(2\boldsymbol{c}(t)) - 1\right), \quad (8)$$

where $t \in \left(t_k, t_{k+1}\right]$, and the elements of $\boldsymbol{c}(t)$ will continue to deterministically decay (at different rates) from $\boldsymbol{c}_{k+1}$ towards target $\bar{\boldsymbol{c}}_{k+1}$.

Specifically, $\boldsymbol{c}_{k+1}$ contains the information of user's previous actions, and the decay rate $\boldsymbol{\delta}_{k+1}$ reflects the influence of the last consumed item on recommendations at time t. Different form traditional LSTM, the updates of $\boldsymbol{c}_{k+1}$, $\bar{\boldsymbol{c}}_{k+1}$ and $\boldsymbol{\delta}_{k+1}$ do not depend on the hidden state from last time-step, but rather on its value $\boldsymbol{h}(t_k)$ at time t_k (after it has decayed for a period of $t_k - t_{k-1}$).

3.3 Modeling Long-Term Demands by Self-Attentive Mechanism

The previous time-aware LSTM addresses the short-term demands of items. However, if an item is consumed long time ago, it can hardly capture the growing influence of demands on the current purchase via time-aware LSTM. Hence we design a special self-attentive component for capturing users' long-term demands. We assume that the periodical purchase demands of products increases as the time goes by. Considering that it is very common that a user may purchase the similar items (items with the same attributes, *e.g.,* category or brand) rather than exactly the same one periodically [40], hence we consider the attribute-level attention of items.

Given a user u and an item i_t with attribute $a_t \in A$ at a future time t. Let $\mathcal{D} \in \mathbb{R}^{|U|\times|A|\times|A|}$ be the estimated purchase time distance matrix of items for all users, $d^u_{a_t,a_{t_j}} \in \mathcal{D}^u$ be the estimated purchase time distance of the current predicting item i_t and all previous item i_{t_j} with attribute a_{t_j}. The diagonal values in matrix $\mathcal{D}^u$ are the learning purchase cycles of products with corresponding attributes, and the non-diagonal values represent the purchase time distance of items with different attributes. Let $\Delta^u_{a_t,a_{t_j}}$ be the observed time interval from purchase history between a_t and the most recent purchase of a_{t_j}. Intuitively, the greater the value $d^u_{a_t,a_{t_j}} - \Delta^u_{a_t,a_{t_j}}$ (indicating a user had just purchased an item with attribute a_{t_j}), the weaker that user u would purchase the same or similar item as i_{t_j}.

We define attentive score α_{t,t_j} to represents the similarity between a previous item and the current predicted one, with their purchase time distance taken into account. The larger attentive score α_{t,t_j} is, the more likely item t_j may be purchased in the current time t. A modified hinge loss $\max\{0, d^u_{a_t,a_{t_j}} - \Delta^u_{a_t,a_{t_j}}\}$ is utilized to model such long-term influence as follows:

$$\alpha_{t,t_j} = \boldsymbol{h}(t_j)^\top \boldsymbol{i}_t - \lambda \log\left(\max\{\gamma, d^u_{a_t,a_{t_j}} - \Delta^u_{a_t,a_{t_j}}\}\right), \quad (9)$$

where $\gamma > 0$, and $\boldsymbol{h}(t_j)^\top \boldsymbol{i}_t$ computes the original similarity of item i_{t_j} and predicted item i_t, λ is a hyper parameter, and $\log(\max\{\gamma, d^u_{a_t,a_{t_j}} - \Delta^u_{a_t,a_{t_j}}\})$ is the penalization for long-term demands.

To take into consideration the influences from all the previous items, we employ an attention mechanism to dynamically select and linearly combine different parts of the hidden representation of input sequence (see Eq. 8) as follows:

$$\boldsymbol{\vartheta}_t = \sum_{j=1}^{n} \frac{\exp\left(\alpha_{t,t_j}\right)}{\sum_{q=1}^{n} \exp\left(\alpha_{t,t_q}\right)} \boldsymbol{h}(t_j), \quad (10)$$

where the $\boldsymbol{\vartheta}_t$ is the attentive weighted sum of $\boldsymbol{h}(t_j)$ for $j \in [1, n]$.

3.4 The Loss Function for Optimization

Given the purchase history $I_{t_n} = \{i_{t_1}, i_{t_2}, ..., i_{t_j}, ..., i_{t_n}\}$ of a user u, our goal is to maximize the *log-likelihood* ℓ of observing items in $I^u_{t_n}$, which can be defined as:

$$\ell(I^u_{t_n}; \theta) = \sum_{j=1}^{n} \log Pr(i_{t_j} | I^u_{t_j}, \Delta t_j), \quad (11)$$

$$= \underbrace{\sum_{j=1}^{n} \log \lambda_{i_{t_j}}(t_j; \theta)}_{\text{purchase}} - \underbrace{\sum_{i_{neg}\in I} \int_{t_1}^{t_n} \lambda_{i_{neg}}(t) dt}_{\text{non-purchase}}, \quad (12)$$

$$= \sum_{i_{neg}\in I} \sum_{j=1}^{n} \left(\frac{1}{|I|} \log \lambda_{i_{t_j}}\left(t_j; \theta\right) - \int_{t_{j-1}}^{t_j} \lambda_{i_{neg}}(t) dt \right),$$

where $\Delta t_j \stackrel{\text{def}}{=} t_j - t_{j-1}$ is the time interval, and $Pr(i_{t_j} | I^u_{t_j}, \Delta t_j)$ is the probability of item i being purchased at time t_j. The first term in Eq. 12 corresponds to the probability of purchase. The second term represents the probability that the item is not purchased in the infinitesimally wide interval $[t, t + dt)$. The above formula is originally proposed in [23]. On can find more details about its derivation from that reference.

3.5 Relationship with Existing Sequential Recommendation Tasks

We formulate our continuous-time recommendation as a generalized sequential recommendation problem (*i.e.,* $\{i_{t_1}, i_{t_2}, ..., i_{t_j}, ..., i_{t_n}\} \rightarrow i_{t_{n+\epsilon}}$): both ordering and time interval information of items are considered. By discrediting time information, our CTRec can easily reproduce next-item and next-session/basket recommendation tasks. For the next-session/basket recommendation (*i.e.,* $\{i_1, i_2, ..., i_j, ..., i_n\} \rightarrow i_{n+\epsilon}$), we can obtain the most likely purchased item in next-basket recommendation by

$$i_{n+\epsilon} = \arg\max_i \int_{t_n}^{t_{n+\epsilon}} \frac{\lambda_i(t; \theta)}{\sum_{j\in I} \lambda_j(t; \theta)} p_i(t; \theta) dt, \epsilon \in \mathbb{N}^*. \quad (13)$$

When the time intervals for the items in next-session/basket $\{t_{n+1} - t_n, t_{n+2} - t_{n+1}, ..., t_{n+\epsilon} - t_{n+\epsilon-1}\}$ are set to the real time spans, our CTRec degenerates to next-session/basket recommendation scenario. For next-item recommendation (*i.e.,* $\{i_1, i_2, ..., i_j, ..., i_n\} \rightarrow i_{n+1}$), we can simply set $\epsilon = 1$.

4 EXPERIMENTS

We evaluate our CTRec model on four real-world datasets. We compare it to several state-of-the-art sequential models showing

Table 1: Statistics of the Datasets.

Dataset	#Users	#Items	#Trans.	#Category	#Co-Pur.	#Re-Pur.
Ta-Feng	26,333	23,736	817,741	2,010	217,908	54.85%
Taobao	19,327	27,152	111,523	2,163	9,153	13.95%
Amazon	45,117	90,996	708,587	65	64,375	82.52%
JingDong	456,974	55,071	8,889,653	51	1,654,611	99.88%

the superiority of our model for sequential recommendation tasks, including next-item and next-session/basket tasks.

4.1 Experimental Settings

4.1.1 Datasets. We experiment with four real-world datasets available to us: Ta-Feng[4], Taobao[5], Amazon[6] and JingDong [43, 44]. The statistics of these four data sets are described in Table 1. Ta-Feng [37] is a Chinese grocery store transaction data from November 1st, 2000 to February 28th, 2001. Taobao, is a user-purchase data (only purchase records are utilized) obtained from Taobao platform[4]. Amazon [11] is a review dataset, *i.e.,* purchase records are collected from the product reviews. We use review records in six months (*i.e.,* from January 1st, 2014 to June 30th, 2014). JingDong contains the purchase records in a quarterly (*i.e.,* from October 1st, 2013 to December 31th, 2013) from one of the largest e-commerce websites in China. Since it is unreliable to include users with few purchase times or limited active time for evaluation, we only keep those whose purchase times are above certain threshold, for example, the thresholds are set to 5, 5, 20, 10 for the 4 datasets respectively in our experiments. Different thresholds are used according to the size of the datasets.

We first study whether short- and long-term purchase behaviors exist in the datasets. For the short-term purchase patterns, we calculate the number of two items that co-occur at least twice within a time window of five items; for the long-term repeated demands, we calculate the percentage of users who has repurchase behavior, *i.e.,* at least one item has been repurchase five times. As shown in Table 1, in the 4 datasets, the number of co-occurred items (#Co-Pur.) are respectively 217908, 9153, 64375, and 1654611; the percentages of users with repurchased demands (#Re-Pur.) are 54.85%, 13.95%, 82.52%, 99.88%. The statistics show clearly that there indeed exist quite a number of co-purchase patterns, and over 50% percentage (except for Taobao dataset) users have repurchase demands of products. This provides strong evidence in support of our approach.

4.1.2 Compared Methods. We compare our model with the state-of-the-art methods from different types of recommendation approaches, including:

BPR [28]: It optimizes the MF model with a pairwise ranking loss. This is a state-of-the-art model for item recommendation, but the sequential information is ignored in this method.

FPMC [29]: It learns a transition matrix based on underlying Markov chains. Sequential behavior is modeled only between the adjacent transactions.

RRN [38]: This is a representative approach that utilizes RNN to learn the dynamic representation of users and items in recommender systems.

NARM [20]: This is a state-of-the-art approach in personalized session-based recommendation with RNN models. It uses attention mechanism to determine the relatedness of the past purchases in the session for the next purchase. As our datasets do not have explicit information of sessions, we simulate sessions by the transactions within each day.

STAMP [21]: STAMP uses a recent action priority mechanism to simultaneously learn from the users' general interests and the current interests.

RMTPP [7]: RMTPP employs a temporal point process as intensity function, and a recurrent neural network is designed to automatically learn a representation of influence from the event history.

Time-LSTM [45]: It designed specifically time gates within LSTM to model time intervals in the sequence. It captures users' short-term and general interests by the time gates in LSTM.

CTRec: our model CTRec utilizes a demand-aware hawkes process framework (DHP) to model the purchase sequence in continuous time. A convolutional neural network and a self-attentive cycles component is designed in DHP to capture the short- and long-term demands of users respectively. To verify the effect of different components, we conduct experiments on the degenerated CTRec models as follows:

- CTRec (T): Only time interval information is utilized (see time-aware LSTM in Sec. 3.2). It is equivalent to the neural Hawkes process model in [23].
- CTRec (S+T): It captures the short-term demands of users by modeling the local sequence information within a convolutional window.
- CTRec (L+T): A self-attentive mechanism is utilized to capture the repeated purchase information of long-term demands from whole purchase history.
- CTRec (S+L+T): Our integrated CTRec model for modeling both short- and long-term demands of users.

The above methods cover different kinds of the approaches in recommender systems: BPR is a classical method among traditional recommendation approaches; FPMC is representative methods which utilize the adjacent sequential information. RRN, NARM and STAMP are methods using the whole sequential information for recommendation. RMTPP and Time-LSTM are recent methods in which time-interval information is considered in RNN. Our CTRec is a continuous-time demand-aware model: the DHP framework with two components for long- and short-term demands. Table 2 summarizes the properties of different methods.

4.1.3 Evaluation Metrics. Given a user, we infer the item that the user would probably buy at a future time. Each candidate method will produce an ordered list of items, we adopt two widely used metrics in sequential recommendation tasks: Hit ratio at rank k (Hit@k) and Normalized Discounted Cumulative Gain at rank k (NDCG@k). Given the predicted ordered list of items at a certain

[4] http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Ta-Feng
[5] https://tianchi.aliyun.com/datalab/dataSet.html?dataId=649
[6] http://jmcauley.ucsd.edu/data/amazon/

Table 2: Properties of methods. P: personalized? N: deep neural network model? S: sequential information ? T: time aware? D: demands aware ?

	BPR	FPMC	RRN	NARM	STAMP	RMTPP	Time-LSTM	CTRec
P	√	√	√	√	√	√	√	√
N	×	×	√	√	√	√	√	√
S	×	√	√	√	√	√	√	√
T	×	×	×	×	×	√	√	√
D	×	×	×	×	√	×	×	√

time point for a user, Hit@k and NDCG@k are defined as

$$Hit@k = \sum_{c=1}^{k} \mathbb{I}(i_c, u), \tag{14}$$

$$NDCG@k = \sum_{c=1}^{k} \frac{\mathbb{I}(i_c, u)}{\log(c+1)}, \tag{15}$$

where c is the position of items in the ranking list. $\mathbb{I}(i_c, u)$ returns 1 if i_c was adopted by user u in original dataset, and 0 otherwise.

Recall that the proposed CTRec is a continuous-time model, we can predict the items being purchased at any feature time. Hence it would be ideal to evaluate our model in a way consistent with continuous-time evaluation, i.e. to predict a purchase at any point in time. However, there are no specific evaluation metric and datasets for the evaluation scenario. We find that the next-session/basket recommendation scenario can be seen as a restricted setting: one session/basket contains a set of discrete time points to be evaluated, each time point corresponds to an item. We evaluate our candidate models at each of those time points separately to simulate our proposed continuous-time recommendation scenario. For the baselines methods which are not time-aware, they can only generate the same ranking list at any time point, hence we use the average evaluation results of the same ranking list at each time points as the final results. We report the average of Hit@k and NDCG@k at all the time points in one session/basket as the final results. We consider the top K (*i.e.,* $K = 5$ and $K = 10$) items in the ranking list as the recommended set and report the average of Hit@k and NDCG@k at all the time points in one session/basket.

4.1.4 Parameter Settings. For each baseline method, grid search is applied to find the optimal settings. These include latent dimensions H from $\{50, 100, 200\}$, and the learning rate from $\{0.1, 0.01, 0.001\}$. We report the result of each method with its optimal hyperparameter settings on the validation data. In our CTRec model, we set the dimensions of latent vector to $[50, 50, 100, 200]$, the window size of filter is set as $[1, 3, 5, 10]$ and learning rate is $[0.01, 0.01, 0.1, 0.1]$. λ in Eq. 9 is set to 0.1. For all the datasets, in next-item recommendation task, we take the last item of each user as the target to predict, the penultimate item as the validation data for model selection. In next-session task, due to the fact that there do not exist session partition, we take the last five percent data as the testing session, the penultimate 5 percent items as the validation data for model selection, and the remaining part in each sequence as the training data to optimize the model parameters.

4.2 Main Results

We present the results of Hit@k and NDCG@k, (*i.e.,* $K = 5$ and $K = 10$) on the next-item, next-session and continuous-time recommendation tasks in Table 3. The results are quite consistent in the three tasks. We have the following observations:

(1) BPR performs better than Pop, but is not as good as FPMC, which uses adjacent sequential information of the transition cubes. This shows that the local adjacent sequential information is useful in predicting next-item.

(2) RRN, NARM and STAMP perform better than BPR and FPMC that do not use neural network (with the exception of RRN model on Hit@k on Ta-Feng dataset). This suggests that neural network is more capable of modeling complex interactions between user's general taste and their sequential behavior. NARM and STAMP perform better than RRN (except for JingDong dataset), with comparable performance, which may lie in that using of attention mechanism helps model to capture the current main purpose of users.

(3) The time-aware models, *i.e.,* RMTPP and Time-LSTM, perform better than the FPMC and BPR, but are less effective than the sequential models, *i.e.,* NARM and STAMP without considering time information. Although RMTPP and Time-LSTM are time-aware models, both have some limitations: RMTPP focuses on learning an event representation vector by considering the temporal point process from event history; while Time-LSTM modifies different time gates in LSTM to control the information learned in the next time step. Both of them are designed to model the event streams to cope with time-sensitive influence from past history. They do not contain the useful characteristic of products and user interests, which are important in e-commerce. The RMTPP model is designed to learn an event representation, rather than event prediction. Therefore, it yields a worse performance than Time-LSTM.

(4) Our continuous-time model CTRec significantly outperforms all the baseline methods on four datasets. Our degenerated model CTRec(T) performs better than another time-aware model Time-LSTM. This implies that the time-aware LSTM with a decay rate over time in our model is more effective than the architecture with time gate in Time-LSTM. Besides, the degenerated models with either short-term or long-term demands information, CTRec(S+T) and CTRec(L+T), are both better than the CTRec(T) that only considers the time information. This indicates that both long-term and short-term purchase demands are useful in predicting items. Particularly, comparing CTRec(T) with CTRec(S), it can be observed that the long-term demands, *i.e.,* purchase cycles of items, is more powerful for recommendation tasks. Our CTRec(S+L+T) model, which considers both long- and short-term demands performs the best, and it significantly outperforms the best baselines.

4.3 Experimental Analysis

4.3.1 Attribute-Level Purchase Cycles. As aforementioned in Sec. 3.3, our continuous-time model CTRec enables us to learn the purchase cycles of products. Recall that the purchase time distance tensor $\mathcal{D}$ holds all the estimated purchase time distance among all items with different attributes. The purchase cycle of a product is the diagonal value of the corresponding attribute in purchase time distance matrix $\mathcal{D}^u$ for a user u. Here we take the product category as the target attribute, and conduct experiments on Amazon and JingDong

Table 3: Performance comparison of different methods.

Datasets	Models	Next-Item Recommendation				Next-Session/Basket Recommendation				Continuous-Time Recommendation			
		Hit@5	Hit@10	NDCG@5	NDCG@10	Hit@5	Hit@10	NDCG@5	NDCG@10	Hit@5	Hit@10	NDCG@5	NDCG@10
Ta-Feng	BPR	0.0539	0.0791	0.0400	0.0480	0.1557	0.2111	0.0629	0.0713	0.0341	0.0240	0.0224	0.0135
	FPMC	0.0554	0.0684	0.0400	0.0441	0.1605	0.2093	0.0684	0.0751	0.0356	0.0241	0.0246	0.0144
	RRN	0.0546	0.0707	0.0416	0.0466	0.1600	0.2041	0.0633	0.0715	0.0339	0.0227	0.0224	0.0131
	NARM	0.0701	0.0944	0.0484	0.0563	0.1756	0.2366	0.0765	0.0863	0.0383	0.0264	0.0271	0.0159
	STAMP	0.0656	0.0778	0.0487	0.0520	0.1791	0.2442	0.0807	0.0903	0.0394	0.0277	0.0286	0.0169
	RMTPP	0.0528	0.0575	0.0393	0.0408	0.1628	0.2119	0.0700	0.0773	0.0351	0.0232	0.0246	0.0141
	Time-LSTM	0.0583	0.0723	0.0404	0.0450	0.1618	0.2138	0.0708	0.0803	0.0337	0.0234	0.0239	0.0140
	CTRec (T)	0.0628	0.0740	0.0443	0.0478	0.1736	0.2447	0.0716	0.0841	0.0388	0.0277	0.0270	0.0162
	CTRec (S+T)	0.0710	0.0807	0.0577	0.0609	0.1828	0.2535	0.0838	0.0975	0.0400	0.0289	0.0273	0.0165
	CTRec (L+T)	0.0762	0.0836	0.0496	0.0516	0.1921	0.2695	0.0815	0.0985	0.0408	0.0297	0.0270	0.0165
	CTRec (S+L+T)	**0.0826***	**0.1189***	**0.0590***	**0.0658***	**0.2075***	**0.2785***	**0.0907***	**0.1050***	**0.0443***	**0.0309***	**0.0303***	**0.0180***
Taobao	BPR	0.0682	0.0909	0.0474	0.0548	0.2972	0.3911	0.1894	0.2182	0.0611	0.0410	0.0423	0.0245
	FPMC	0.0752	0.0975	0.0536	0.0607	0.3006	0.3742	0.1989	0.2202	0.0617	0.0388	0.0445	0.0248
	RRN	0.0908	0.1085	0.0617	0.0674	0.3117	0.3865	0.2064	0.2297	0.0645	0.0405	0.0460	0.0257
	NARM	0.1020	0.1264	0.0715	0.0795	0.3439	0.4507	0.2095	0.2419	0.0708	0.0472	0.0476	0.0276
	STAMP	0.1145	0.1248	0.0676	0.0727	0.3734	0.4820	0.2321	0.2676	0.0773	0.0518	0.0517	0.0301
	RMTPP	0.0867	0.1109	0.0598	0.0678	0.3297	0.4709	0.2050	0.2499	0.0679	0.0494	0.0455	0.0277
	Time-LSTM	0.0976	0.1064	0.0537	0.0606	0.3321	0.4733	0.1977	0.2421	0.0681	0.0496	0.0448	0.0274
	CTRec (T)	0.0993	0.1196	0.0675	0.0741	0.3534	0.4904	0.2122	0.2539	0.0726	0.0510	0.0474	0.0284
	CTRec (S+T)	0.1147	0.1245	0.0696	0.0766	0.3709	0.5040	0.2281	0.2689	0.0774	0.0539	0.0515	0.0307
	CTRec (L+T)	0.1179	0.1287	0.0742	0.0771	0.3861	0.5296	0.2339	0.2789	0.0794	0.0556	0.0530	0.0316
	CTRec (S+L+T)	**0.1347***	**0.1436***	**0.0792***	**0.0814***	**0.3996***	**0.5307***	**0.2427***	**0.2835***	**0.0821***	**0.0557***	**0.0543***	**0.0319***
Amazon	BPR	0.0077	0.0130	0.0054	0.0072	0.0388	0.0469	0.0201	0.0221	0.0082	0.0049	0.0056	0.0031
	FPMC	0.0084	0.0102	0.0056	0.0061	0.0431	0.0577	0.0226	0.0264	0.0091	0.0061	0.0059	0.0035
	RRN	0.0110	0.0122	0.0077	0.0080	0.0549	0.0652	0.0287	0.0314	0.0114	0.0068	0.0075	0.0042
	NARM	0.0127	0.0149	0.0082	0.0085	0.0810	0.1043	0.0432	0.0499	0.0173	0.0114	0.0112	0.0065
	STAMP	0.0131	0.0176	0.0074	0.0082	0.0828	0.1172	0.0437	0.0534	0.0175	0.0126	0.0114	0.0069
	RMTPP	0.0102	0.0120	0.0061	0.0065	0.0915	0.1338	0.0478	0.0586	0.0190	0.0140	0.0121	0.0075
	Time-LSTM	0.0112	0.0135	0.0070	0.0076	0.0938	0.1177	0.0490	0.0563	0.0197	0.0127	0.0124	0.0071
	CTRec (T)	0.0130	0.0160	0.0073	0.0079	0.1073	0.1522	0.0556	0.0689	0.0226	0.0169	0.0140	0.0088
	CTRec (S+T)	0.0135	0.0188	0.0076	0.0088	0.1124	0.1572	0.0580	0.0707	0.0233	0.0167	0.0143	0.0088
	CTRec (L+T)	0.0147	0.0183	0.0066	0.0074	0.1129	0.1480	0.0573	0.0677	0.0236	0.0159	0.0146	0.0087
	CTRec (S+L+T)	**0.0188***	**0.0218***	**0.0089***	**0.0095***	**0.1243***	**0.1778***	**0.0620***	**0.0776***	**0.0261***	**0.0194***	**0.0160***	**0.0100***
JingDong	BPR	0.0341	0.0385	0.0161	0.0177	0.0300	0.0350	0.0101	0.0110	0.0060	0.0035	0.0035	0.0019
	FPMC	0.0320	0.1040	0.0162	0.0403	0.0335	0.0440	0.0105	0.0129	0.0067	0.0049	0.0038	0.0024
	RRN	0.0385	0.043	0.0169	0.0185	0.0345	0.0730	0.0117	0.0194	0.0071	0.0076	0.0043	0.0035
	NARM	0.0500	0.1020	0.0239	0.0400	0.0375	0.0505	0.0094	0.0118	0.0075	0.0051	0.0033	0.0021
	STAMP	0.0517	0.0625	0.0265	0.0312	0.0340	0.0430	0.0105	0.0134	0.0079	0.0056	0.0039	0.0025
	RMTPP	0.0395	0.1205	0.0214	0.0472	0.0325	0.0360	0.0152	0.0158	0.0065	0.0036	0.0056	0.0029
	Time-LSTM	0.0560	0.0580	0.0306	0.0313	0.0380	0.0520	0.0099	0.0129	0.0078	0.0054	0.0035	0.0023
	CTRec (T)	0.0505	0.0885	0.0276	0.0392	0.0390	0.0540	0.0104	0.0137	0.0082	0.0058	0.0038	0.0025
	CTRec (S+T)	0.0555	0.0870	0.0292	0.0390	0.0430	0.0645	0.0123	0.0170	0.0086	0.0070	0.0045	0.0031
	CTRec (L+T)	0.0575	0.0950	0.0309	0.0426	0.0435	0.0915	0.0142	0.0231	0.0093	0.0099	0.0051	0.0042
	CTRec (S+L+T)	**0.0665***	**0.1055***	**0.0358***	**0.0484***	**0.0590***	**0.0845***	**0.0213***	**0.0265***	**0.0126***	**0.0092***	**0.0076***	**0.0048***

" $*$ " indicates the statistically significant improvements (*i.e.,* two-sided t-test with $p < 0.05$) over the best baseline.

datasets[7] to get a sense of purchase cycles of products. We present the average purchase cycles of all users in Fig. 2. An observation is that the purchase cycles of *"Beauty/Makeup"* and *"Office"* products in Amazon and JingDong datasets have relatively short purchase cycles, while for *"Book"* in Amazon and *"Health Care Equipment"* in JingDong dataset, the purchase cycles are relatively large. All these make sense since *"Book"* and *"Health Care Equipment"* seem far more durable than the consumption of products in the categories of *"Beauty/Makeup"* and *"Office"*.

More insight can be obtained if more fine-grained categories (nearly to item-level) are used to observe purchase cycles of products. We will leave this to future work. In our current work, we utilize the existing categories in the datasets, and give an overall learning of products purchase cycle in a certain category.

[7]It is difficult to see without a description of the category information available. For better explanation, we conduct analysis experiments only on Amazon and JingDong datasets, in which we can obtain the name of categories rather than an ID in Ta-Feng and Taobao datasets.

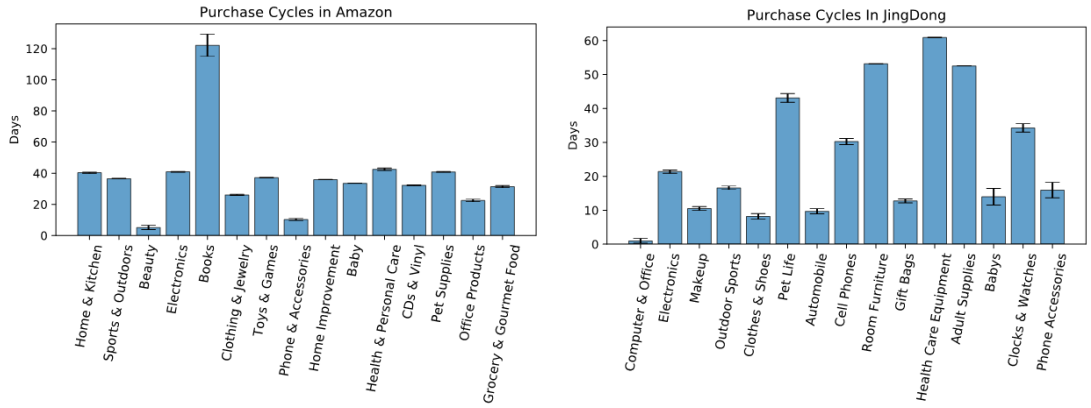

Figure 2: Purchase cycles on category-level in Amazon and JingDong datasets

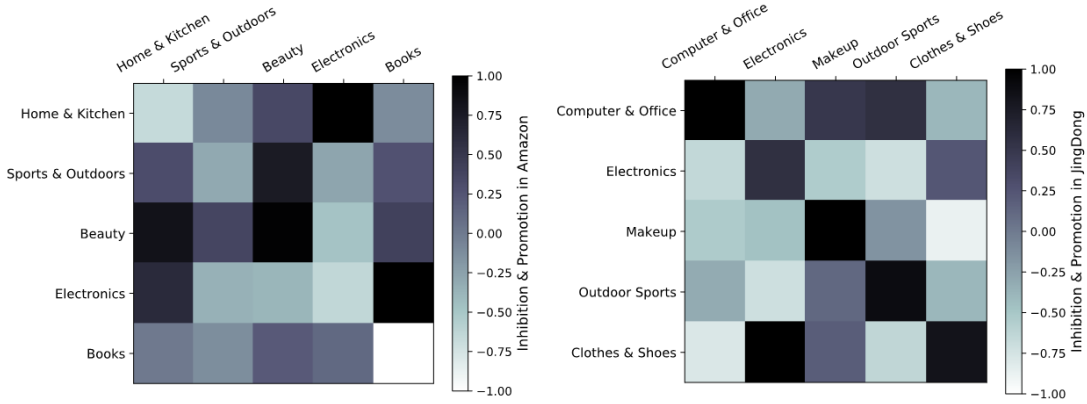

Figure 3: Promotion and inhibition influence among different product categories.

4.3.2 Purchase Dependency among Categories. Purchase demands of products are also influenced by purchases of other categories. The dependency between purchases in different categories can be measured with the non-diagonal elements in purchase time distance tensor $\mathcal{D}$. Given a category c_m and c_n, d_{c_m,c_n} is the purchase time distance of c_n after purchasing c_m, hence the promotion or inhibition score are computed by

$$d_{mean} = \frac{\sum_{k=1}^{|A|} d_{c_k}}{|A|}, \tag{16}$$

$$Score(c_m, c_n) = \frac{-(d_{c_m,c_n} - d_{mean})}{(\sum_{k=1}^{|A|} |d_{c_m,c_n} - d_{mean}|)/|A|}, \tag{17}$$

where the smaller the purchase time d_{c_m,c_n}, the greater the promotion score between categories c_m and c_n.

A negative score means inhibition influence of c_m on c_n, and a positive value means promotion impact. As shown in Fig. 3, the color in each cube represents the influence, which is scaled in the right color bar. Taking the cube with position "Beauty → Electronics" for example, "→" means the influence on "Electronics" after purchasing "Beauty". We can observe that: the category "Beauty" and "Electronics" in both Amazon and JingDong datasets inhibits the purchasing of each other, which may due to the portrait of adopters, *e.g.,* the makeup products adopter is more likely a women who maybe somewhat less interested in electronic products.

4.3.3 Repurchase Time Prediction. Given a user, the repurchase time of items can be computed according to Eq. 5. It's a item-level purchase time learned by our model. We first analysis the accuracy of our model on Amazon dataset. By setting a time window with a range of window size, *i.e.,* {1 day, 5 days, 10 days, 20 days}, if the predicted repurchase time falls within the same window as the real purchase time, we set the accuracy score to 1, otherwise 0. We present the accuracy of CTRec in Table 4. To make a comparison, we make a naive prediction: we compute the average of the repurchase time of all products (*i.e., Avg_time*), and predict that all items are repurchased at the average time. It can be observed that our model CTRec can make much more accurate predictions than the method with average time; the comparison with CTRec(T) also demonstrates the effectiveness of leveraging long- and short-term demands.

As for "toilet addict case" introduced in Sec 1, we can leverage two kinds of information to avoid the phenomenon: (1) the average repurchase time learned by CTRec of the item " toilet seat cushion" is 46 days; (2) the purchase cycle of its category "Health & Personal

Table 4: The Accuracy of Predicting Repurchase Time

Dataset	Model	1 Day	5 Days	10 Days	20 Days
Amazon	Avg_Time	0.0066	0.0407	0.0952	0.2960
	CTRec(T)	0.0115	0.0540	0.1223	0.4470
	CTRec	0.0301	0.1409	0.3148	0.7391

Care" in Amazon dataset is 42 days (see in Fig. 2). Both information indicate reasonable recommendation cycles and can be utilized to avoid the successive recommendation in a short time.

5 RELATED WORK

Sequential recommender systems have attracted a lot of attention from the research community and industry. According to the way they use the time information, we summarize the related methods of sequential recommender systems as follows.

General Sequential Methods. Many approaches have been proposed to detect the purchase appetites of users and their evolution over time. They have been applied in different recommendation scenarios: next-item, next-basket or session-based recommendation tasks. For the *next-item recommendation* task, sequential-based approaches directly model the transaction of items in the sequence [1, 6, 10, 18, 21, 31, 34, 38]. Transaction-based model captures the short-term interests of users by modeling users as translation vectors operating on item sequences [10]. To better capture the short-term interests, convolutional filters are utilized in [32] to learn local sequential patterns in top-N recommendation. Similarly, a mixture model with CNN and RNN is used in LSTNet [18] to extract short-term local dependency patterns and discover long-term patterns for time series treads. In addition to using RNN to capture the sequential information, some methods based on sequential patterns are also proposed to extract the co-occurrences (or dependencies) of items or periodical characteristic of item purchases [8, 24, 35, 39]. The *next-basket recommendation* aims at predicting a set of items the user could put in his basket [2, 29, 37, 41]. For example, Morkov Chains (MC) based methods, *e.g.,* the Factorizing Personalized Markov Chains (FPMC) [29], the RNN based models, *e.g.,* Hierarchical Representation Model (HRM) [37]. *Session-based recommendation* models [13, 15, 20, 21, 26] are commonly used to predict web page clicking. This is different from next basket recommendation in that the order of clicks on items in a session is also considered. In these models, users' preference are learned by the clicked items in the sessions [13, 26]. To make more accurate prediction, attention mechanism has been utilized in [20, 21] to capture user's main interests in the current session. In most of the above sequential methods, they treat users' general interests as the long-term interests, and the dependencies of items in sequence as the short-term interests. In our CTRec model, in addition to the above elements, we also capture repeated purchases as the long-term demands. Moreover, all the previous studies only consider the sequential order of object, while ignoring the time interval information in the sequence. In our study, we showed that this is an important factor to consider for modeling users' behavior.

Time-Sensitive Sequential Methods. Some recent studies have proved that time intervals between users' actions are of significant importance in capturing users' actions and for which the traditional RNN architectures are insufficient [14, 17, 22, 30, 36, 42, 45]. For example, the Time-LSTM model in [45] equips LSTM with time gates to model time intervals: the specific time gates enables model capture users' short-term and long-term interests. Similarly, in RNN-based next Point-Of-Interest recommendation [42], different distance gates are designed to control short- and long-term interest updates. Another way to integrate time interval information is formulating the dynamic dependence of items in the sequence as a point process (*e.g.,* Hawkes process), in which the streams of discrete events in the past are modeled in continuous time [5, 14, 23, 36]. For example, the extended point process model with a hierarchical RNN architecture in [36] is a session-level model, which only leverages the time intervals between sessions. Such session-based model may lose the temporal information within the session. Besides, with the aim of learning representations of users and items, the point process can also be utilized to model dynamic embeddings of users and items from a sequence of user-item interactions by recurrent model [5, 17]. The most related work to our model is [23], in which a neural hawkes process model allows past events to influence the future in complex and realistic ways, by conditioning future event intensities on the hidden state of a recurrent neural network. However, with consideration of the purchase demands of users, the incentive effect may increase with a certain cyclicality for long-term demands while decrease in the short-term demands, and such influence cannot be handled in the simple event streams model in [23].

6 CONCLUSION

In this paper, we argue that meeting users' purchase demands at the right time is one of the key factors for e-commerce recommendation, yet has largely been ignored in the literature. We propose a continuous-time recommendation model based on demand-aware hawkes process to address user's long-term and short-term demands adaptively. The proposed model not only is capable of learning the purchase cycles of products within each category, but also captures the temporal influence among products in different categories. Compared with previous methods, the ability of making accurate prediction on repurchase time enables our model to avoid the common recommendation failures, such as the case of "toilet addict case[2]". Additionally, the proposed model can be easily adapted to general sequential recommendation tasks, such as next-item and next-session/basket recommendation. As the future plan, we will make more elaborate category classification, so as to conduct more detailed experiments on capturing and understanding the influence of the purchase cycles of products on item-level.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China under Grant No. 18XNLG22, the Science and Technology Project of Beijing under Grant No. Z181100003518001, an NSERC Discovery grant and IVADO. We thank Yulong Gu for his insightful comments and discussions.

REFERENCES

[1] Ting Bai, Pan Du, Wayne Xin Zhao, Ji-Rong Wen, and Jian-Yun Nie. 2019. A Long-Short Demands-Aware Model for Next-Item Recommendation. *arXiv preprint arXiv:1903.00066* (2019).

[2] Ting Bai, Jian-Yun Nie, Wayne Xin Zhao, Yutao Zhu, Pan Du, and Ji-Rong Wen. 2018. An Attribute-aware Neural Attentive Model for Next Basket Recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1201–1204.

[3] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. 2017. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1979–1982.

[4] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy It Again: Modeling Repeat Purchase Recommendations. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 62–70.

[5] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[6] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-based Recurrent Neural Network Recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 152–160.

[7] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1555–1564.

[8] Riccardo Guidotti, Giulio Rossetti, Luca Pappalardo, Fosca Giannotti, and Dino Pedreschi. 2017. Next Basket Prediction using Recurring Sequential Patterns. *arXiv preprint arXiv:1702.07158* (2017).

[9] Sung Ho Ha, Sung Min Bae, and Sang Chan Park. 2002. Customer's time-variant purchase behavior and corresponding marketing strategies: an online retailer's case. *Computers & Industrial Engineering* 43, 4 (2002), 801–820.

[10] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 161–169.

[11] Ruining He and Julian McAuley. 2016. 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*. International World Wide Web Conferences Steering Committee, 507–517.

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 549–558.

[13] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 241–248.

[14] Seyedabbas Hosseini, Ali Khodadadi, Keivan Alizadeh, Ali Arabzadeh, Mehrdad Farajtabar, Hongyuan Zha, and Hamid RR Rabiee. 2018. Recurrent poisson factorization for temporal recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).

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

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

[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. [n. d.]. Learning Dynamic Embeddings from Temporal Interaction Networks. *Learning* 17 ([n. d.]), 29.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 95–104.

[19] Patrick J Laub, Thomas Taimre, and Philip K Pollett. 2015. Hawkes processes. *arXiv preprint arXiv:1507.02822* (2015).

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

[21] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1831–1839.

[22] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1015–1024.

[23] Hongyuan Mei and Jason M Eisner. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*. 6754–6764.

[24] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 669–672.

[25] Andrzej Nowak and Robin R Vallacher. 1998. *Dynamical social psychology*. Vol. 647. Guilford Press.

[26] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[27] Seyyed Mohammadreza Rahimi and Xin Wang. 2013. Location recommendation based on periodicity of human activities and location categories. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 377–389.

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

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

[30] Hee Seok Song. 2018. Deep Neural Network Models to Recommend Product Repurchase at the Right Time. *Journal of Information Technology Applications & Management* 25, 2 (2018), 73–90.

[31] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-rate deep learning for temporal recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 909–912.

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

[33] C-Y Tsai and C-C Chiu. 2004. A purchase-based market segmentation methodology. *Expert Systems with Applications* 27, 2 (2004), 265–276.

[34] Xuan-An Tseng, Da-Cheng Juan, Chun-Hao Liu, Wei Wei, Yu-Ting Chen, Shih-Chieh Chang, and Jia-Yu Pan. 2018. Nested LSTM: Modeling Taxonomy and Temporal Dynamics in Location-Based Social Network. (2018).

[35] Petre Tzvetkov, Xifeng Yan, and Jiawei Han. 2005. TSP: Mining top-k closed sequential patterns. *Knowledge and Information Systems* 7, 4 (2005), 438–457.

[36] Bjørnar Vassøy, Massimiliano Ruocco, Eliezer de Souza da Silva, and Erlend Aune. 2018. Time is of the Essence: a Joint Hierarchical RNN and Point Process Model for Time and Item Predictions. *arXiv preprint arXiv:1812.01276* (2018).

[37] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 403–412.

[38] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 495–503.

[39] Ghim-Eng Yap, Xiao-Li Li, and Philip S Yu. 2012. Effective next-items recommendation via personalized sequential pattern mining. In *International Conference on Database Systems for Advanced Applications*. Springer, 48–64.

[40] Jinfeng Yi, Cho-Jui Hsieh, Kush R Varshney, Lijun Zhang, and Yao Li. 2017. Scalable Demand-Aware Recommendation. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2412–2421. http://papers.nips.cc/paper/6835-scalable-demand-aware-recommendation.pdf

[41] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 729–732.

[42] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Zhixu Li, Jiajie Xu, and Victor S Sheng. 2018. Where to Go Next: A Spatio-temporal LSTM model for Next POI Recommendation. *arXiv preprint arXiv:1806.06671* (2018).

[43] Wayne Xin Zhao, Sui Li, Yulan He, Edward Y Chang, Ji-Rong Wen, and Xiaoming Li. 2016. Connecting social media to e-commerce: Cold-start product recommendation using microblogging information. *IEEE Transactions on Knowledge and Data Engineering* 28, 5 (2016), 1147–1159.

[44] Xin Wayne Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. 2014. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1935–1944.

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