Time Matters: Sequential Recommendation with Complex Temporal Information

Wenwen Ye[1], Shuaiqiang Wang[2], Xu Chen[3**], Xuepeng Wang[2], Zheng Qin[1], Dawei Yin[4]
[1]Tsinghua University, [2]Data Science Lab, JD.com, [3]University College London, [4]Baidu Inc.
[1]{yeww14, qingzh}@mails.tsinghua.edu.cn, [2]{shqiang.wang@gmail.com, wangxuepeng1@jd.com},
[3]successcx@gmail.com, [4]yindawei@acm.org

ABSTRACT

Incorporating temporal information into recommender systems has recently attracted increasing attention from both the industrial and academic research communities. Existing methods mostly reduce the temporal information of behaviors to behavior sequences for subsequently RNN-based modeling. In such a simple manner, crucial time-related signals have been largely neglected. This paper aims to systematically investigate the effects of the temporal information in sequential recommendations. In particular, we firstly discover two elementary temporal patterns of user behaviors: "absolute time patterns" and "relative time patterns", where the former highlights user time-sensitive behaviors, e.g., people may frequently interact with specific products at certain time point, and the latter indicates how time interval influences the relationship between two actions. For seamlessly incorporating these information into a unified model, we devise a neural architecture that jointly learns those temporal patterns to model user dynamic preferences. Extensive experiments on real-world datasets demonstrate the superiority of our model, comparing with the state-of-the-arts.

KEYWORDS

Recommender systems, Temporal information, Sequential user behaviors, E-commerce

ACM Reference Format:
Wenwen Ye[1], Shuaiqiang Wang[2], Xu Chen[3**], Xuepeng Wang[2], Zheng Qin[1], Dawei Yin[4]. 2020. Time Matters: Sequential Recommendation with Complex Temporal Information. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401154

1 INTRODUCTION

Recommender system, as an effective information filtering method, has been deployed in many real applications, ranging from video sharing website, e-commerce to on-line bookstore and social network. For capturing user dynamic preferences, recent years have witnessed an emerging trend on modeling user sequential behaviors [14, 18], and the basic assumption lies in that "a user's former behaviors may influence her latter ones".

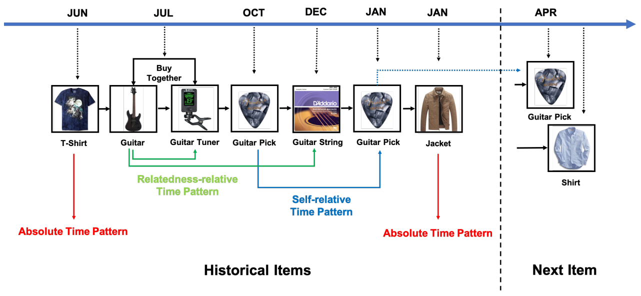


Figure 1: An example of leveraging user historical records for recommendation.

Existing sequential recommendation algorithms mostly focus on the sequences of users' behaviors while ignoring their temporal patterns [41, 43]. For example, traditional [17] recommender systems typically model the temporal evolution of the user interests using concept drifting with the assumption that users' interests evolve smoothly over time. Recent methods based on Neural Networks, especially the session-based recommendations [14, 18], mainly focus on tracking short-term user interests and only take the sequential orders of the items into modeling. Beyond that, some sequential recommendation models embed the bucketized absolute time and positions with self-attention [15, 41]. Though there are some initial efforts on integrating temporal information into sequential recommendations, they mostly take incomplete (or partial) temporal information into consideration. There is still a lack of efforts on comprehensively and thoroughly modeling such integrated temporal patterns.

In this study, we systematically investigate the effects of the temporal information in sequential recommendations. As exampled in Figure 1, a user repeatedly purchased clothes such as a t-shirt in June and a jacket in January. Though a rough category of "clothes" can be used to summarize her interest, a better refined model can also infer the associated seasonal information, e.g., the user would be interested in shirts in April. Such temporal patterns provide distinct evidences to obtain users' time sensitive preferences in recommendation. Notably, another important observation is that the user initially purchased a guitar along with a guitar tuner, and then bought guitar picks and strings several months later. Zooming into the temporal details of the purchase sequences, temporal patterns emerge as follows: when a user purchases a guitar, she might not be interested in guitar picks or strings immediately, but would

** Corresponding author.

SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401154

buy them in about three and five months respectively; furthermore, guitar picks are periodically purchased every three months.

According to these observations, two typical temporal patterns of user behavior sequences can be recognized: the **Absolute Time Pattern** and the **Relative Time Pattern**. Absolute Time Patterns directly relate users' behavior sequences to the timestamp: in the above example, the user's purchase of clothes is directly related to time and varies with seasons. Relative Time Patterns emphasize the time intervals between behaviors and can be further divided into two subcategories: Self-relative Time Pattern and Relatedness Relative Time Pattern. Self-relative Time Patterns, for example, the consecutive repurchases of guitar picks every three months characterize periodical behaviors in e-commerce, which typically applies to consumables, e.g. diapers, beers, etc. By contrast, Relatedness Relative Time Patterns add the time dimension into consideration by revealing the relatedness relationship between items. In Figure 1, the interval between between guitar and strings can be determined as five months because initially both of them were purchased in July while the string replacement occurred in December.

To tackle these issues, we propose a novel neural network framework named **T**emporal **A**ttentive **SE**quential **R**ecommendation (TASER), which can jointly learn those temporal patterns and user interests for sequential recommendations. For each input sequence, an absolute temporal module that contains a group auto-encoder embedding network is introduced to embed the time sequences. This module enriches the capacity for representing the absolute time, compared with traditional time bucketization methods. Together with the absolute temporal module, a relative temporal module injects the relative temporal effects into the relationships between two actions of user behavior sequences in an explicit manner. Moreover, a time-aware constraint is adopted in our loss function to obtain a better representation of the temporal information. Finally, the effectiveness of our proposed sequential recommendation model is verified through extensive experiments on two large-scale real-world datasets.

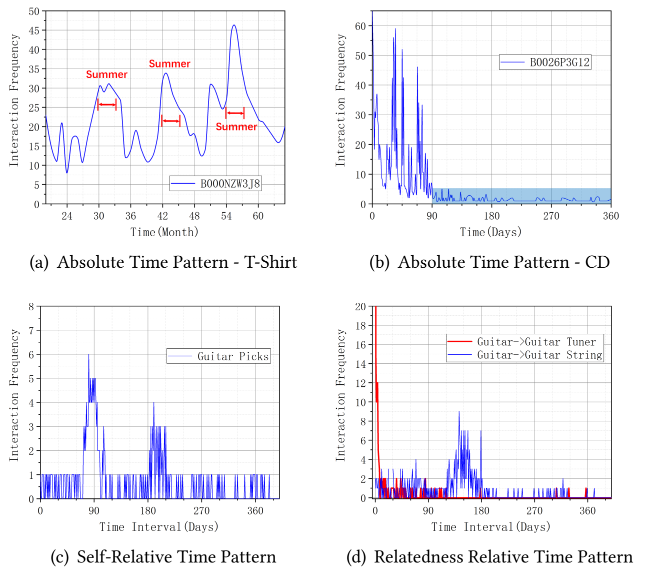
Our main contributions are summarized as follow:

- We discover absolute time patterns and relative time patterns based on insightful data analysis to model users' temporal behaviors for recommender systems.
- We propose a novel time-aware neural network — TASER to leverage such temporal patterns in a unique perspective.
- We conduct extensive experiments on real-world datasets to demonstrate the significant performance gains of our model against state-of-the-art methods.

The rest of paper is organized as follows: In section 2, we first conduct preliminary analysis based on a real-world dataset to verify our assumption and formulate our problem formally. We then propose our model TASER in section 3. In sections 4 and 5, we intensively compare our TASER with state-of-the-art baselines over seven public datasets.Finally the related works and conclusions of this study are presented in sections 6 and 7, respectively.

2 PROBLEM FORMULATION

In this section, we present several case studies to verify the key concepts of temporal patterns in sequential recommendation.



(a) Absolute Time Pattern - T-Shirt (b) Absolute Time Pattern - CD (c) Self-Relative Time Pattern (d) Relatedness Relative Time Pattern

Figure 2: Cases of various time patterns. (a) and (b) illustrate two absolute time patterns, while (c) and (d) demonstrate the self-relative and relatedness relative time patterns, respectively. Note that the "Summer" months in (a) include June, July and August, and the blue shadow in (b) means that the sales per day is under five.

2.1 Case Study

All of our analysis is based on the Amazon datasets[1] [13], which involves user purchasing behaviors as well as the corresponding timestamps ranging from May 1996 to July 2014.

2.1.1 Absolute Time Pattern. The absolute time pattern is a pointwise concept, characterizing the absolute timestamp as a unique variable in depicting user-item interactions. Timestamp can provide the context and auxiliary information to determine whether a user is interested in an item, which is useful in revealing periodical behaviors. As exampled in Figure 2(a), the sales of a T-shirt (Item ID in the dataset: B000NZW3J8) sharply rised from May to September every year, which is consistent with the common knowledge that customers are more likely to buy T-shirts in summer than in winter. Another example is the daily sales of a music album released on Nov 23, 2009, which is shown in Figure 2(b). The sales of the music album have a strong relationship with the release time. In the beginning, the sales were relatively high and stable, and then declined as time passed, and eventually fell under 5 after 90 days.

2.1.2 Relative Time Pattern. The relative time pattern is a pairwise concept, which focuses on the time interval between each pair of user behaviors. Intuitively, such information reflects the influence of the former behavior upon the latter one, and different pairs may exhibit various patterns. We further propose two kinds of relative time patterns: Self-relative time patterns and Relatedness relative time patterns. Figure 2(c) shows an example of a self-relative

[1]http://jmcauley.ucsd.edu/data/amazon/

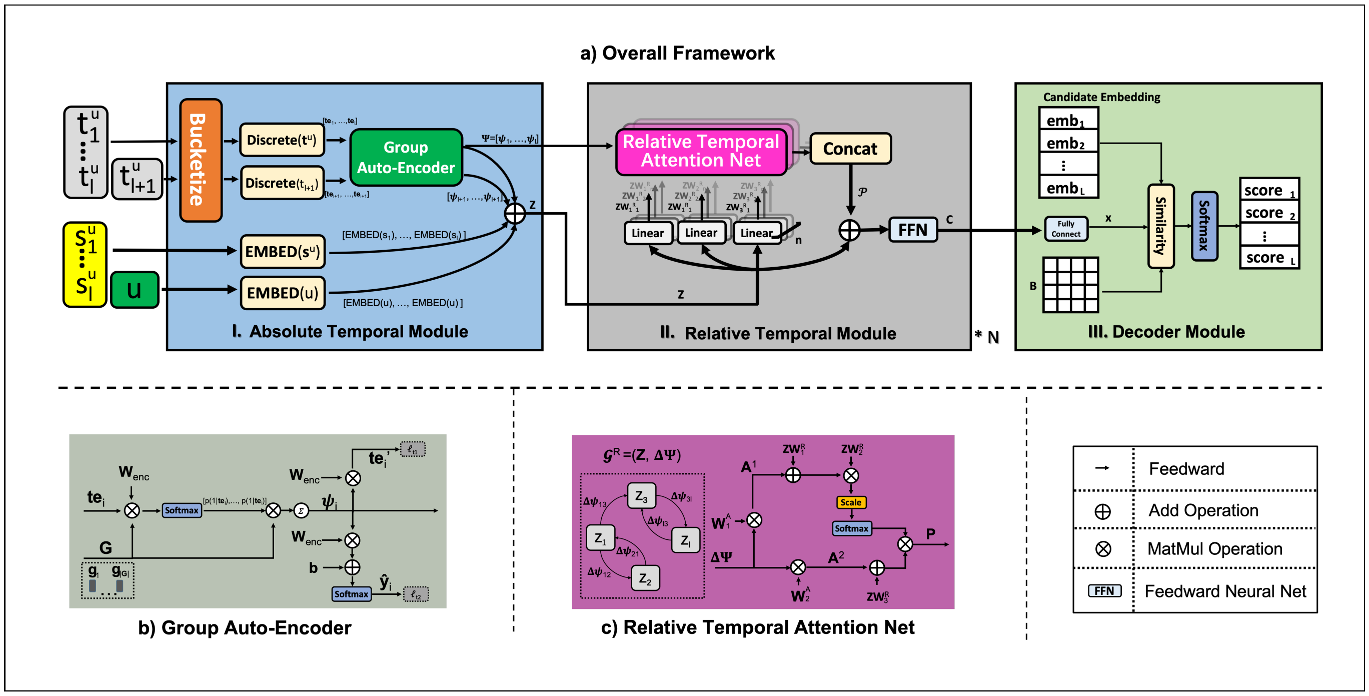


Figure 3: Illustration of our temporal-aware model. (a) The overall framework. (b) The detailed implementation of Group Auto-encoder structure. (c) Left: The labeled, directed, fully-connected graph consisting of sequence elements and their relative time. Right: The detailed implementation of Relative Temporal Attention Net.

time pattern. The green line presents a user's repeated purchases of guitar picks, where the intensity of purchases is a function of time and $x = 0$ means the first purchase. Figure 2(d) illustrates two relatedness relative time patterns on item pairs guitar->tuner (red line) and guitar->strings (blue line) respectively. Most users purchase guitars and tuner together, while typically buy new strings about 150 days later than the guitar. This observations demonstrate two specific time intervals between purchases of the two item pairs, which are important relatedness relative time patterns for recommendations.

From the above case studies, we can conclude that both categories of time patterns are essential to accurately model user behaviors in practical recommender systems.

2.2 Problem Definition

In this paper, we represent matrices, vectors, and scalars as bold capital letters (e.g., $\boldsymbol{X}$), bold lower-case letters (e.g., $\boldsymbol{x}$), and normal lower-case letters (e.g., x), respectively. Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{I}$. For a user $u \in \mathcal{U}$, we chronologically organize her purchasing behaviors as $S^u = [(s_1^u, t_1^u), (s_2^u, t_2^u)..., (s_{l_u}^u, t_{l_u}^u)]$, where the pair (s_i^u, t_i^u) means a user u interacted with an item s_i^u at time t_i^u, and $t_1^u \le t_2^u \le ... \le t_{l_u}^u$ holds. The set of all user behaviors is defined as $\mathcal{S} = \{\boldsymbol{S^u} \mid u \in \mathcal{U}\}$. Then the task of sequential recommendation with temporal information can be formally defined as follows. Given $\mathcal{U}, \mathcal{I}, \mathcal{S}$ and the target timestamp t, our task aims to predict the target item for each user on which the user would conduct her behavior at t.

3 OUR MODEL

In this section, we present TASER, a temporal attentive sequential recommendation model. The architecture of our model is shown in the top of Figure 3. First of all, the **absolute temporal module** embeds the sequential behaviors with timestamps into dense vectors, which are then fed into the **relative temporal module** to generate the semantic vectors of the target user' historical behaviors. Then the similarity between the target user and the candidate items can be calculated for prediction in the **decoder module**.

3.1 Absolute Temporal Module

The absolute temporal module is designed to capture the absolute time information. The input of this module includes four components: the sequence of items $\boldsymbol{s}^u$, the corresponding sequence of time $\boldsymbol{t}^u$, the target user u and the next interaction time t_{l+1}^u. The output is $Z = [z_1, z_2, ...z_l]$, where an arbitrary element $z_i \in \mathbb{R}^d$ is a d-dimensional vector:

$$z_i = \mathrm{E}(u) + \mathrm{E}(s_i^u) + \mathrm{E}(t_i^u) + \mathrm{E}(t_{l+1}^u) \tag{1}$$

where $\mathrm{E}(\cdot)$ represents the embedding layer, which directly encodes discrete variables into embedding, and discretizes continuous variables into discretized values for encoding [5]. Formally, $\mathrm{E}(t_i^u) = \boldsymbol{te}_i = \mathrm{E}(Bucketize(t_i^u))$, where $Bucketize(\cdot)$ is a function that bucketizes time into discrete values, and $\boldsymbol{te}_i \in \mathbb{R}^d$ is a d-dimensional embedding vector of the time t_i^u.

The granularity of the bucketization may vary quite differently, and inappropriate bucketization usually results in poor performances. To address this issue, we integrate group auto-encoder [30] into this module. Suppose that there are a set of latent groups of vectors $\boldsymbol{G} = \{\boldsymbol{g}_1, \boldsymbol{g}_2, ..., \boldsymbol{g}_{|G|}\}$, where the cardinality $|G|$ is smaller than the number of the bucket size T_{Bucket}, i.e,. $|G| < T_{Bucket}$ holds. The goal of a group auto-encoder is to encode the time embedding $\boldsymbol{te}_i$ into a probability distribution over $\boldsymbol{G}$. As we can see in Figure 3(b), the probability that $\boldsymbol{te}_i$ is mapped to the jth group can be calculated as follows:

$$p(j \mid \boldsymbol{te}_i) = \frac{\exp(\boldsymbol{te}_i^\top \boldsymbol{W}_{enc}\, \boldsymbol{g}_j)}{\sum\limits_{k\in|G|} \exp(\boldsymbol{te}_i^\top \boldsymbol{W}_{enc}\, \boldsymbol{g}_k)} \tag{2}$$

where $\boldsymbol{g}_j, \boldsymbol{g}_k \in \mathbb{R}^d$ represents the embedding of the ith group $g_i \in G$, and $\boldsymbol{W}_{enc} \in \mathbb{R}^{d\times d}$ is the encoding matrix that computes proximity between $\boldsymbol{te}_i$ and $\boldsymbol{g}_j$. Then the group embedding of $\boldsymbol{te}_i$ can be calculated as follows:

$$\boldsymbol{\psi}_i = \sum_{j=1}^{|G|} p(j \mid \boldsymbol{te}_i)\, \boldsymbol{g}_j, \quad \boldsymbol{te}_i' = \boldsymbol{W}_{dec}\, \boldsymbol{\psi}_i, \tag{3}$$

where $\boldsymbol{\psi}_i$ can be seen as the high-level representation of $\boldsymbol{te}_i$. Then the output of the Absolute Temporal Embedding module can be rewritten as follows:

$$\boldsymbol{z}_i = \mathrm{E}(u) + \mathrm{E}(s_i^u) + \boldsymbol{\psi}_i + \boldsymbol{\psi}_{l+1} \tag{4}$$

The loss ℓ_a of this auto-encoder involves two terms ℓ_{t_1} and ℓ_{t_2}:

$$\begin{aligned} \ell_a &= \ell_{a_1} + \ell_{a_2}, \\ \ell_{a_1} &= ||\boldsymbol{te}_i - \boldsymbol{te}_i'||, \\ \ell_{a_2} &= \sum_{m=1}^{T_{Bucket}} -\boldsymbol{y}_i[m] \log \hat{\boldsymbol{y}}_i[m] \end{aligned}, \tag{5}$$

where ℓ_{a_1} formulates the difference between the embedding $\boldsymbol{te}_i$ and the group embedding $\boldsymbol{te}_i'$ of the time t_i^u, and ℓ_{a_2} calculates the cumulative log loss of the predictions. In particular, $\boldsymbol{y}_i$ is the one-hot representation for the ground truth of the ith bucketized time, and $\hat{\boldsymbol{y}}_i = \mathrm{softmax}\,(\boldsymbol{W}_t\ \boldsymbol{\psi}_i + \boldsymbol{b}_t)$ is its prediction, where $\boldsymbol{W}_t \in \mathbb{R}^{T_{Bucket}\times d}$.

3.2 Relative Temporal Module

As mentioned in Section 2, the pairwise relationship between each pair of items has a strong connection with the time interval between the purchases of the items. For a sequence with l items, the embedding matrix of the relative time is $\Delta\boldsymbol{\Psi}$, where the element $\Delta\boldsymbol{\psi}_{ij}$ is in the ith row and the jth column, and $1 \le i \le l, 1 \le j \le l$, $\Delta\boldsymbol{\psi}_{ij} = \boldsymbol{\psi}_i - \boldsymbol{\psi}_j$ and $\boldsymbol{\psi}_i$,$\boldsymbol{\psi}_j$ are shown in Equation (3).

Currently, Recurrent Neural Networks (RNN) is the most effective category of methods for sequential recommendations. However, such methods [43] only consider the relative time between adjacent items. They generally ignore the cases between non-adjacent ones, which could be useful for predictions as well. Thus the dependencies between any pair of non-adjacent items can be only estimated along the full path between the two, resulting in certain estimation loss. As the sequence increases, such loss grows sharply, which might significantly deteriorate the accuracy of these methods [28].

In this paper, we formulate the sequential items in users' sequential behaviors as a labeled, directed, fully-connected graph $\mathcal{G} = (Z, E)$, where the set of the nodes Z represents the items, and the set of the edges E includes all of the pairwise sequences from the head nodes to the tails. Thus $\Delta\boldsymbol{\Psi}$ is the embedding of E, where the element $\Delta\boldsymbol{\psi}_{ij}$ is the embedding of the edge from the ith node to the jth one. Since $\mathcal{G}$ is directed and its adjacency matrix is asymmetric, it is impossible to directly generate a corresponding Laplacian matrix and conduct a spectral decomposition on it for calculations. Inspired by the non-spectral approaches [29], we introduce an attention-based architecture named Relative Time Attention Net, which can propagate the semantic information of the nodes and the edges in $\mathcal{G}$.

As we can see in figure 3(c), let $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ be two predicted adjacency matrices of $\mathcal{G}$, which can be formulated as follows:

$$\boldsymbol{A}^1 = \Delta\boldsymbol{\Psi}\ \boldsymbol{W}_1^A, \quad \boldsymbol{A}^2 = \Delta\boldsymbol{\Psi}\ \boldsymbol{W}_2^A,$$

where $\boldsymbol{W}_1^A, \boldsymbol{W}_2^A \in \mathbb{R}^{|\Delta\boldsymbol{\psi}_{ij}|\times|\Delta\boldsymbol{\psi}_{ij}|}$ are the trainable weight matrices to distill useful information for propagation. The relative temporal attention net can be divided into three steps: Attention coefficient calculation, attention coefficient normalization, and final representation refinement, which are presented as follows.

For each input sequence $\boldsymbol{Z} = [z_1, z_2, \cdots, z_l]$, where $z_i \in \mathbb{R}^d$ is the embedding of the item node z_i outputted by the absolute temporal module, we firstly calculate the attention coefficient matrix $\boldsymbol{E} \in \mathbb{R}^{l\times l}$, a.k.a., attention map, based on the query, key and relative time. Particularly, the coefficient e_{ij} between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ indicates the importance of the node $\boldsymbol{z}_j$ to $\boldsymbol{z}_i$ when their relative time is $\Delta\boldsymbol{\psi}_{ij}$, which can be formulated as:

$$e_{ij} = \frac{(z_i \boldsymbol{W}_1^R)(z_j \boldsymbol{W}_2^R + (\boldsymbol{\psi}_i - \boldsymbol{\psi}_j)\boldsymbol{W}_1^A)^\top}{\sqrt{|z_i \boldsymbol{W}_1^R|}}, \tag{6}$$

where $\boldsymbol{W}_1^R$ and $\boldsymbol{W}_2^R \in \mathbb{R}^{d\times d}$ are the trainable transformation matrices. The denominator $\sqrt{|z_i \boldsymbol{W}_1^R|}$ is the parameter to scale the dot product attention. We assume that $z_i\boldsymbol{W}_1^R, z_i\boldsymbol{W}_2^R$ are independent with mean 0 and variance 1, and the dot $z_i\boldsymbol{W}_1^R \cdot z_i\boldsymbol{W}_2^R$ has mean 0 and variance $|z_i\boldsymbol{W}_1^R|$.

To make coefficients easily comparable across different nodes, we normalize them across all nodes with a softmax function such that for each node z_i, the cumulative coefficients from z_i to all of the nodes in Z is equal to 1. Then the normalized attention coefficients are leveraged to calculate the final output representations $\boldsymbol{p}_i$ for each node z_i. Here $\boldsymbol{p}_i \in \mathbb{R}^d$ is a d-dimension vector, which is a linear combination of the features corresponding to z_i's first-order neighbors and their relative time. Formally,

$$\boldsymbol{p}_i = \sum_{j=1}^{l} e_{ij} \left(z_j \boldsymbol{W}_3^R + (\boldsymbol{\psi}_i - \boldsymbol{\psi}_j)\, \boldsymbol{W}_2^A \right) \tag{7}$$

where $\boldsymbol{W}_3^R \in \mathbb{R}^{d\times d}$ is a trainable transformation matrix.

We employ multi-head operation [28] to stabilize the learning process of attention, which has been proved is helpful to improve the performance through different representation sub-spaces. In particular, we split $\boldsymbol{W}_1^R, \boldsymbol{W}_2^R, \boldsymbol{W}_3^R$ into n parts, and then execute n independent relative temporal attention nets for the sequence of the item node embeddings $\boldsymbol{Z}$. Then we concatenate the n outputs into $\boldsymbol{\mathcal{P}} = [\boldsymbol{P}_1, \cdots, \boldsymbol{P}_n]$. Here the ith element $\boldsymbol{P}_i$ is the output of the

ith relative temporal attention net, a matrix that is composed of the representations for all of the nodes. Note that here the relative time information has been propagated into each node representation. By doing this, the output of the whole Relative Temporal Module can be formulated as follows:

$$\boldsymbol{C} = \mathrm{FFN}(\boldsymbol{Z} + \boldsymbol{\mathcal{P}}) = \mathrm{ReLU}\left((\boldsymbol{Z} + [\boldsymbol{P}_1; \boldsymbol{P}_2; \cdots ; \boldsymbol{P}_n])\, \boldsymbol{W}^C\right) + \boldsymbol{b}, \quad (8)$$

where the output matrix $\boldsymbol{C} \in \mathbb{R}^{|S^u| \times d}$, the trainable matrix $\boldsymbol{W}^C \in \mathbb{R}^{d \times d}$ and the biased vector $\boldsymbol{b} \in \mathbb{R}^d$. Here FFN$(\cdot)$ is a feed-forward network with a ReLU activation. By residual connecting $\boldsymbol{\mathcal{P}}$ with the original input $\boldsymbol{Z}$, we propagate the low-level features into a higher level. After a stack of N identical relative temporal module blocks, we receive $\boldsymbol{C}_N$ as the next downstream network's input with the same size as $\boldsymbol{Z}$.

To summarize, the relative temporal attention net can make full use of all the edges associated with relative time to make predictions, and the relationship between each pair of items can be calculated directly. Note that conventional RNN-based methods only calculate the dependency between successive items, and the dependencies between non-adjacency items cannot be modeled straightforwardly, which may result in unsatisfactory results.

3.3 Decoder Module

Then we employ a fully-connected layer to transform $\boldsymbol{C}_N$ into a vector $\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^d$. We then need a downstream network to predict the probability that each candidate item should be returned or not with x. Most of existing methods [41] utilize a fully-connected layer to determine the scores. Thus they need at least $d \times m$ parameters, where m is the number of the candidate items. Since usually m is a very large number, we have to reserve a large space to store these extra parameters, which could increase the time and space costs significantly. In this study, we adopt an alternative bi-linear decoding scheme [18] to compute a similarity score between the representations of the current user and each candidate item, which is formulated as follows:

$$score_i = \mathrm{E}(s_i^u)^\top \, \mathbf{B} \, \boldsymbol{x}, \quad (9)$$

where $\mathbf{B} \in \mathbb{R}^{d \times d}$ is a trainable matrix, and $\mathrm{E}(\boldsymbol{I}_i)$ is the candidate item's embedding. This approach can significantly reduce the number of the parameters from $d \times m$ to $d \times d$, where d is usually much smaller than m. Then we softmax these similarity score to acquire the probability for each item to be purchased.

3.4 Loss Function

Self-attention [28] just leverages the position information of the sequence. State-of-the-art methods like ATRank [41] directly encode the absolute time information for embedding. None of them can consider the relative time information (time intervals), which could be extremely important in sequential recommendations. Our time-related loss function is designed as follows.

$$\ell_t = \sum_{i=1}^{l} \frac{\|\boldsymbol{\psi}_{l+1} - \boldsymbol{\psi}_i\|_2}{|t_{l+1} - t_i|} \quad (10)$$

where $||\cdot||_2$ is the L2 norm of a vector, and here $\|\boldsymbol{\psi}_{l+1} - \boldsymbol{\psi}_i\|$ calculates the distance between the time embeddings $\boldsymbol{\psi}_{l+1}$ and $\boldsymbol{\psi}_i$ from Equation (3). Divided by the time interval between two absolute time stamps, ℓ_A could distinguish same orders of items with different intervals, which provides more accurate time information for sequential recommendations. In addition, we also use the negative logarithm of likelihood to form a binary cross-entropy loss:

$$\ell_r = \sum_u \sum_{i \in I_u^+} -\log(score_i^u) + \sum_u \sum_{j \in I_u^-} -\log(1 - score_j^u)), \quad (11)$$

where I_u^+ is the set of u's purchased items (arranged in purchasing orders). Following previous works [4, 27], for each target item i, we randomly sample several (5 in our experiments) negative instances in the second term.

Our total loss $\mathcal{L}$ consists of three terms: the prediction loss ℓ_r, the auto-encoder loss ℓ_a in Equation (5) and the temporal loss ℓ_t in Equation 10, which are combined linearly as follows:

$$\mathcal{L} = \ell_r + \beta \ell_a + \gamma \ell_t, \quad (12)$$

where β and γ are the trade-off parameters. With the temporal loss $\gamma \ell_A$, our model can produce better representations of the temporal information than state-of-the-art methods.

3.5 Discussion

In this section, we show that TASER is a general framework for sequential recommendations. It is essentially a generalization of ATRank [41] and SASRec [15].

ATRank [41] can be viewed as a special case of TASER without the temporal mechanisms such as the absolute time group auto-encoder, the relative time module and the temporal loss function. In ATRank, the encoding of the input is simply $\boldsymbol{z_i} = \mathrm{E}(u) + \mathrm{E}(s_i^u) + \boldsymbol{te}_i + \boldsymbol{te}_{l+1}$. By removal of the relative time information in TASER, the coefficient e_{ij}, the final representation $\boldsymbol{p}_i$ and the loss in such variant of TASER can be rewritten as follows:

$$e_{ij} = (z_i \boldsymbol{W}_1^R)(z_j \boldsymbol{W}_2^R)^\top, \;\; \boldsymbol{p}_i = \sum_{j=1}^{l} \alpha_{ij}(z_j \boldsymbol{W}_3^R), \;\; \mathcal{L} = \ell_r$$

which is the same as ATRank.

SASRec [15] did not utilize any temporal information. The encoding of its input is $\boldsymbol{z_i} = \mathrm{E}(u) + \mathrm{E}(s_i^u) + \mathrm{E}(i) + \mathrm{E}(l+1)$. The representations of the attention mechanism and the loss function for SASRec are the same as ATRank. Thus SASRec is generalized by TASER as well.

4 EXPERIMENTAL SETTINGS

Our experiments aim to answer the following research questions:

RQ1 Does the proposed model achieve the state-of-the-art performance in sequential recommendations?

RQ2 What are the effects of various components in the TASER architecture?

RQ3 How does the temporal information of the sequence affect the recommendation results?

RQ4 What are the effects of the key hyper-parameters?

4.1 Datasets

We conduct our experiments on the following datasets:

- **LastFM**[2]**.** It is a widely used music recommendation data. We employ the 1K version in our experiments.

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

Table 1: Statistics of the evaluation datasets. Inter. represents the number of interaction, Avg. means the average interaction per user, Spa.(%) is the sparsity of these datasets, and Rep.(%) calculates the proportion of repeat purchases, which are the interactions indicating self-related time patterns.

dataset	Users	Items	Inter.	Avg.	Spa.(%)	Rep.(%)
LastFM	943	12105	83272	88.305	0.729	7.633
Video	1000	3296	15849	15.849	0.481	0.520
Instrument	2129	20928	55382	26.013	0.120	3.270
Beauty	8934	27762	202307	22.644	0.304	2.621
Music	2893	13183	64320	22.233	0.169	4.520
Phone	11567	34086	344724	29.802	0.064	0.725
Cloth	39387	23033	278677	7.0756	0.323	1.930

- **Amazon.** This dataset contains a collection of user-item interactions on Amazon spanning from May 1996 to July 2014. We evaluate our model and its variants on six product categories, including Instant Video, Musical Instrument, Beauty, Digital Music and Cell Phone. For sequential recommendations, we select the users with at least 10 purchasing records and the items with at least 5 times of interactions for experiments.

The statistical details of these datasets are presented in Table 1, including the numbers of the users, items, interactions, average interactions per user (Avg. Inter. for short in the table), and the sparsity of the dataset. In particular, the sparsity of the dataset is defined as the ratio of the known ratings (ground-truth) in the data matrix.

Following [4, 15, 27], we take the first 70% of interactions in each user's purchase sequence as the training set and use the next 10% of interactions as the validation set to search the optimal hyper-parameter settings for all models. The remaining 20% interactions in each user's sequence are used as the test set for evaluating a model's performance.

4.2 Evaluation Metrics

Let $\hat{R}_u^{1:N}$ be a list of top N predicted items for user u, where $\hat{R}_u^i$ is ranked at the ith position in $\hat{R}_u$ based on its predicted score. Let R_u be the test data with totally M users in our system. The following metrics are used to evaluate the recommendation results:

F1-score: We adopt per-user averaging instead of global averaging to achieve better interpretability [16]:

$$\begin{aligned} \mathrm{P}@N &= \frac{1}{M}\sum_{M}\frac{|\hat{R}_u^{1:N}\cap R_u|}{N} \\ \mathrm{R}@N &= \frac{1}{M}\sum_{M}\frac{|\hat{R}_u^{1:N}\cap R_u|}{|R_u|} \\ \mathrm{F}_1@N &= \frac{1}{M}\sum_{M}\frac{2\mathrm{P}_u@N\cdot \mathrm{R}_u@N}{\mathrm{P}_u@N+\mathrm{R}_u@N} \end{aligned} \tag{13}$$

NDCG: The normalized discounted cumulative gain evaluates ranking performance by considering the right positions of correct items, which can be formulated as follows:

$$\mathrm{NDCG}@N = \frac{1}{Z}\mathrm{DCG}@N = \frac{1}{Z}\sum_{i=1}^{N}\frac{2^{r_i}-1}{\log_2(i+1)} \tag{14}$$

where $r_i \in \{1,0\}$ is the ground-truth of the items indicating whether the user purchased the target item or not in the Amazon datasets, and whether the target user listen the song or not in LastFM. Z is a normalization constant that is the maximum possible value of DCG@N.

4.3 Baselines

We compare our method with the following baselines:

- **POP.** This is a non-personalized method, where all of the items are ranked by their popularity for each user.
- **BPR [23].** Bayesian personalized ranking is a state-of-the-art method for non-sequential item recommendation.
- **FPMC [24].** Factorized personalized Markov chains is a state-of-the-art method for sequential recommendation based on Markov chains. For our sequential recommendation problem, each basket represents one item.
- **GRU4Rec [14].** It is a session-based recommendation method that uses RNN structured GRU for sequence recommendations.
- **NARM [18].** It is a session-based recommendation model that integrates an attention mechanism into RNN. It can capture both the users' sequential behaviors and their main purposes in each session.
- **Time-LSTM [43].** It is a variant of LSTM, which can model users' sequential actions, equipping LSTM with time gates to model time intervals.
- **ATRank. [41]** It is an self-attention framework to model user behaviors.
- **TASER-Position.** It is a variant of our TASER, which only uses the position orders of the sequences instead of the temporal information for prediction.
- **TASER-Abs.** It is another variant of the TASER that uses the original self-attention without relative time in the relative temporal module.

4.4 Implementation Details

The implementation details of our method are listed as follows. The dimension d of all embedding latent vectors ($|d_{\mathrm{model}}|$, $|d_S|$, $|d_u|$, $|d_T|$) is determined by grid search in the range of {5, 10, 20, 30, 40, 50, 100} in TASER and all of the baselines (if exists). We adopt $n = 5$ multi-heads and $N = 3$ relative blocks in our relative temporal module. All of the parameters are first randomly initialized according to the uniform distribution, and then updated with stochastic gradient descent (SGD). The learning rate of SGD is determined by grid search in the range of $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. Dropout rate is tuned among {0, 0.3, 0.5, 0.7}. Batch Size is set as 32 for all methods. The least length of the input sequence is set as 2 (including the target item) for all datasets at first and could be extended to 20 maximally. The target length R is set to 1 for all datasets. The number of group for auto-encoder $|G| = 365$. $\beta = 0.1$and $\gamma = 0.01$ to balance different terms in the loss function. All experiments were conducted on 4 NVIDIA Tesla P40 GPU with 24GB of on-board memory. For all baselines, hyper-parameters are tuned with grid search with validation set. In our experiments, each user's purchasing records are ordered in their purchasing time, and the first 70% items of each user are used for training, while the remaining are used for testing. We recommend 5 items (N = 5) for each user.

Table 2: Summary of the performance for baselines and our models. POP and BPR are non-sequential baselines. FPMC, GRU4Rec, NARM, Time-LSTM and ATRank are sequential baselines. The underlined numbers are the results achieved by the best baseline. TASER, TASER-Position and TASER-Abs are our model and its two variants respectively. The bold numbers are the best performance of each column, and all of the numbers in the table are percentages with % omitted.

dataset	LastFM		Video		Instrument		Beauty		Music		Phone		Cloth	
Measure@5(%)	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG	F_1	NDCG
POP	3.474	16.162	1.226	5.106	0.923	5.110	0.776	4.504	1.002	5.230	0.856	4.523	0.622	4.350
BPR	4.833	20.043	1.278	6.112	1.006	4.860	0.807	4.543	1.021	5.317	0.893	4.741	0.636	4.380
FPMC	6.432	21.311	$\underline{1.384}$	$\underline{7.001}$	$\underline{1.032}$	5.087	0.831	$\underline{5.012}$	$\underline{1.074}$	5.349	$\underline{0.969}$	$\underline{5.055}$	0.667	4.463
GRU4Rec	6.628	23.562	1.299	6.832	1.017	5.021	0.825	4.660	1.058	5.345	0.933	4.964	0.654	4.406
NARM	6.723	24.303	1.311	5.903	1.026	5.051	0.826	4.726	1.065	5.354	0.940	5.010	0.661	4.425
Time-LSTM	6.684	23.020	1.309	6.918	1.024	5.063	0.827	4.715	1.067	$\underline{5.360}$	0.939	5.024	0.663	4.406
ATRank	$\underline{6.895}$	$\underline{25.246}$	1.375	6.962	1.030	$\underline{5.095}$	$\underline{0.834}$	4.941	1.071	5.344	0.965	5.045	$\underline{0.696}$	$\underline{4.463}$
TASER-Position	7.312	27.318	1.474	7.206	1.043	5.181	0.895	5.515	1.133	5.484	1.003	5.255	0.728	4.677
TASER-Abs	7.213	26.221	1.462	7.124	1.039	5.137	0.891	5.322	1.120	5.410	0.993	5.183	0.717	4.515
TASER	**7.822**	**29.125**	**1.546**	**7.242**	**1.051**	**5.206**	**0.932**	**5.785**	**1.192**	**5.608**	**1.012**	**5.524**	**0.762**	**4.789**

5 EXPERIMENTAL RESULTS

We firstly report the experimental results of our model against the baselines to answer the research question RQ1, and then conduct comprehensive ablation studies to response the research question RQ2. We then utilize a case study to answer RQ3, and finally analyze the effect of our hyper-parameters for RQ4.

5.1 Overall Performance (RQ1)

- POP and BPR have worse performances than sequential methods due to the loss of sequence information, which evidences sequence information's (e.g. position order and time) important role in recommendation.
- For sequential methods, FPMC and ATRank outperform all other baselines on 6 subcategories of Amazon dataset with $F1$ and $NDCG$, and ATRank and NARM perform the best on LastFM among all the baselines. Such results demonstrate the promising performance of the deep models in sequential recommendations.
- ATRank overcomes NARM on almost all of the datasets, indicating that the pure attention mechanism is more efficient than RNN and attention mechanism fusion methods in sequence recommendation.
- Encouragingly, TASER consistently and noticeably outperforms all other baselines on all of the datasets. These results double confirm that the temporal patterns are extremely important for sequential recommendations.
- TASER achieves better performance than its variants TASER-Position and TASER-Abs, and particularly our two variants can perform better than all baselines on almost all datasets. Such observations demonstrate the effectiveness of our proposed absolute temporal module and relative temporal module for sequential recommendations against conventional RNN-based techniques.
- It is interesting to see that the improvement of our model on LastFM is larger than that on other datasets. The reason can be that LastFM includes more repeat purchases than others. Our model can effectively capture such pattern, resulting in more significant gains on this dataset.

5.2 Ablation Study (RQ2)

In this section, we analyze the impacts of our components via an ablation study to answer RQ2. Table 3 shows the performance of TASER and its 5 variants on LastFM, Video and Instrument datasets in term of $F1@5$. The results on other datasets are similar, and thus we do not report their results due to the page limit. The variants and their experimental results are listed as follows:

- (1) *Use Position* (TASER-Position): As described before, we replace the time information in TASER with the position order. The experimental result demonstrates that the concepts of the absolute and relative time patterns can be also applied to position orders.
- (2) *Remove Relative Position from (1)*: Without the relative position information, the performance significantly drops, which illustrate that the relative information is important to sequential recommendation. This result further supports the conclusion that the relative temporal module can effectively regularize TASER to achieve better performance.
- (3) *Only Use Position Embedding (**SASRec**)*: Missing the absolute time group auto-encoder, relative temporal module and time-related loss function, this variant performs the worst. We further replace the position embedding with the following position encoding Equation 15. The experimental result turns out that the position embedding representation yields no better result than position encoding.

$$\begin{aligned} PE(p, 2i) &= \sin t/10000^{2i/d}, \\ PE(p, 2i+1) &= \cos t/10000^{2i/d} \end{aligned} \tag{15}$$

  Such results demonstrate that the temporal information is indispensable for sequential recommendation against the position orders.
- (4) *Remove Relative Time propagation (TASER-Abs)*: As shown in Tables 2 and 3, the relative temporal information has a great impact on the recommendation results.
- (5) *Remove Temporal Loss Function*: The time-related loss function improves the representation of time embedding significantly. More details on this loss function are presented in Section 6.3.

Table 3: Ablation analysis on three datasets in term of $F1@5$.

dataset	LastFM	Video	Instrument
(0) Default	**7.822**	**1.546**	**1.051**
(1) Use P	7.312	1.474	1.043
(2) Remove RP	7.020	1.402	1.035
(3) Only Use PE	6.886	1.389	1.033
(4) Remove RT	7.213	1.426	1.039
(5) Remove TF	7.247	1.481	1.046

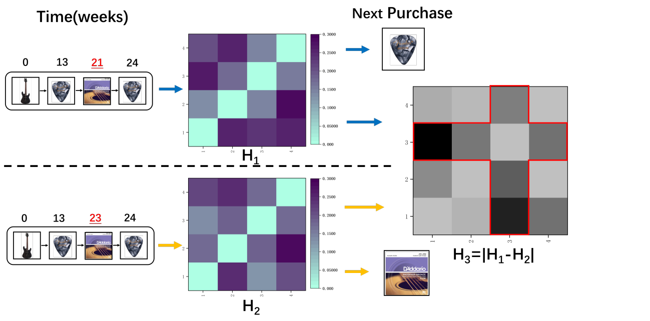


Figure 4: Visualization of two sequences with the same order of actions but different times

5.3 Does Time Affects Sequential Recommendation? (RQ3)

In this part, two case studies on the Amazon-Instrument dataset are designed to clarify this question.

5.3.1 Change the time in the sequence. S_1 and S_2 are two similar sequences. The only difference between the two is that the purchase time of the third item in S_1 and S_2 are 21 and 23 respectively. The attention maps $\mathbf{H_1}$ and $\mathbf{H_2}$ are visualized in the left of Figure 4. It can be clearly observed that the huge discrepancy between the two heat maps are still quite big. If we merely use position order instead of the temporal information for predictions, $\mathbf{H_1}$ and $\mathbf{H_2}$ should be very similar, as their inputs are the same. To further analyze the impact of changing the time of the third item in the sequence, we plotted the difference map associated with the third item, which demonstrates the significant difference between the two. Furthermore, the right side of the figure shows the recommended results for the two sequences simultaneously. TASER captures the internal time variation of the sequence and presents it on the recommendations.

5.3.2 Change the next purchase time. Few sequential recommendation algorithms have studied the impact of different purchase time on recommendations. To answer this question, we design a set of experiments to study the effect of various next purchase time on a sequence. Figure 5 records the top 5 recommendations of the sequence S at the time of 36, 40 and 48. At the time of 36, item 1 ranks the first and item 2 ranks the fourth. As time changes to 40, item 1 falls to the fifth and item 2 rises to the first. When the time is at 48, item 1 returns to the first place and item 2 falls by one place. This result clearly shows that our TASER can recommend different results as the next purchase time passes with the full and quantitative modeling of time.

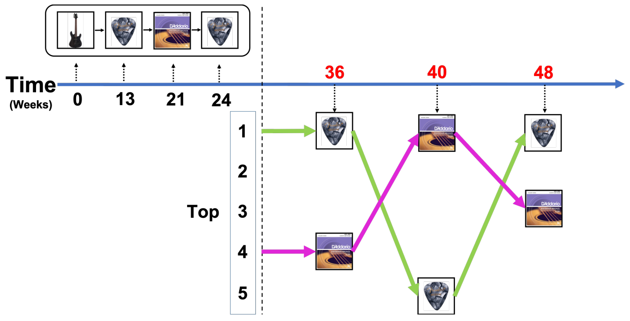


Figure 5: Changes in recommendation results at different purchase times. The red numbers represent the different next purchase time.

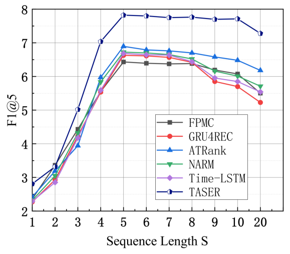
5.4 Model Analysis (RQ4)

In this section, we further analyze the important hyper-parameters and the time-related loss function in our model.

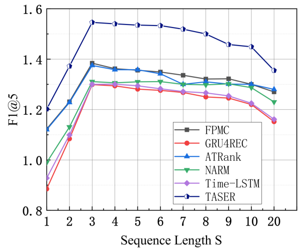
5.4.1 Impact of sequence length. Figure 6 shows the impact of the sequence length on LastFM and Amazon-Video. From the figure we can see that our TASER achieves the best performance with different lengths on the two datasets. For example, the best length setting on LastFM is 5, while the number on Amazon-Video is 3. Furthermore, the length that offers the best performance is positively related to the number of the user average interaction. For example, the average interaction in LastFM is 88, which are much larger than those in the Amazon datasets according to Table 1. Furthermore, GRU4Rec and NARM outperform FPMC on LastFM. This can be explained by the fact that both GRU4Rec and NARM are multiple-step RNN-based methods, and thus are able to acquire more sequential signals from long sequences [27] than the two-step method FPMC. However, GRU4Rec and NARM perform worse than FPMC on Video. Because these non-personalized RNN-based methods can easily get biased on training sets of Amazon and enlarge the generalization error to some extent despite the use of regularization and dropout as described in [14].

It is reasonable to infer that the attention mechanism can capture long-path dependencies between any pair of positions in the sequence, evidencing that TASER and ATRank continuously outperform other methods irrespective of the sequence length. The models are more efficient in learning the features on long sequences because within both models, the maximum length of the paths in the attention map is always 1. TASER outperforms ATRank by at least 5% resulting from the relative temporal module, which positively contributes to long sequence modeling. The fact that TASER still outperforms other methods by at least 5% even when the sequence length is more than 20, which indicates that our method can better capture long-distance dependencies in long sequences.

5.4.2 Impact of Group Number. As discussed in section 3.3, we adopt group auto-encoder to embed the absolute time, and the parameter $|G|$ controls the number of group embedding. Due to the limited resources, instead of traversing, we selected several numbers $|G| = 0, 7, 30, 90, 365, 500$ in our experiments (corresponding to the days in a week, a moth, a quarter and a year), where $|G| = 0$ means

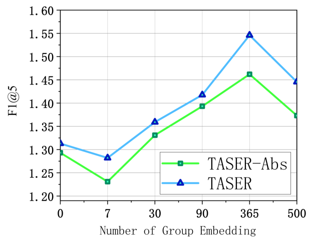


(a) LastFM

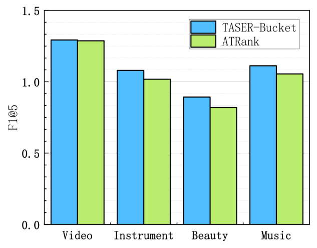


(b) Amazon-video

Figure 6: Performance under different number of sequence length.



(a) Group Embedding



(b) Optimization Function

Figure 7: (a) Performance under different number of $|G|$ (b) Performance comparison with different optimization functions.

it uses the most basic bucketization without group embedding. As shown in Figure 7(a), when $|G| = 7$, our model performs worse than the basic bucketization. As $|G|$ increases, the performance of our model improves, and constantly outperforms the basic one. The effect of group embedding is limited when the number of group is small, which may result from too much noises introduced into embedding. The model's performance achieves climax when $|G| = 365$ and declines after that. The increase of the parameter reduces the performance when it is too large. The experimental results are the best when $|G|$ is equal to 365, the days in a year. It can be concluded that the user's purchase behaviors are related to the actual time period and the group embedding has advantages in dealing with absolute time compared to mere time bucketization.

5.4.3 Impact of Temporal Loss. To determine the effect of the temporal loss function, we design a comparative experiment between ATRank and TASER-Bucket (a TASER variant without the group embedding module for absolute time and relative time matrix). Thus, the only difference between TASER-Bucket and ATRank is that the former has the temporal loss while the latter does not, which is exhibited in Section 3.4. By modeling the temporal information quantitatively in the time-space, as shown in Figure 7(b), the performance significantly improves.

6 RELATED WORK

Our work is essentially an integration of temporal information and sequential recommendation. Below, we present a review of the related work on these two research directions.

6.1 Sequential Recommendation

Many models have been proposed to utilize user history in a sequential manner for future behavior prediction and recommendation. Factorized personalized Markov chains (FPMC) [24] integrates matrix factorization with Markov chains for next-basket recommendation which embeds the adjacent behaviors' information into item latent factors.

In order to model sequential behaviors, [12] propose a sparse sequential recommendation approach with Markov chains and [44] propose a sequential recommender based on Markov chains with probabilistic decision-tree models. [14] and [26] model short-term preferences for session-based recommendations using RNN through previous user clicks and purchase behaviors. [18, 21] further combine attention mechanism and RNN. What is more, [4, 27] use memory networks and CNN [10] for sequential recommendation. Unlike above sequential models that use convolutional or recurrent modules, [28] is purely based on self-attention and [1, 15, 39, 41] extent self-attention into recommendation.

In addition to e-commerce, sequential recommendations have been also applied to a variety of application scenarios, such as music recommendations [3, 11, 33], POI recommendations [7, 9], browsing recommendations [40] and so on. Existing models typically implicitly prioritize the user's previous records into orders based on timestamps, without distinguishing the different roles that the time information may play in predicting current interests. However, in this work, we use the temporal attention network to store and manipulate each user's previous interactions, which helps to enhance the expressiveness of sequence recommendation.

6.2 Temporal Information in Recommendation

Meanwhile, temporal information is an important contextual information for recommendation. Both [20] and [42] mention the importance of time recommendations in e-commerce platforms.

In collaborative filtering-based methods, modeling temporal information has been already employed in [6, 17, 34, 36] to prove its susses. [19] uses the temporal information of micro-blogs to find semantically and temporally relevant topics in order to profile user interest drifts with an implicit information network. [2] provides explicit continuous-time random process models of the creation of users and topics, and of the evolution of their interests. [37] investigates the temporal dynamics of user interests in tagging systems and propose a temporal interests model for tracking users' interests. [35] proposes to design a tensor factorization approach to capture the temporal dynamics of users' preferences over time which treats time as an additional dimension. [8, 32] have expanded the latent factor based models to characterize the dependency and transition between users' current latent vector via social networks.

Recently, recurrent network-based methods achieve a great success for many temporal tasks. Some pioneering works [22, 25, 31, 38] adopt RNN for temporal recommendations. [43] uses the time interval to explore relationship of adjacent objects and [41] integrates time as a feature into the model by bucketize timestamp. Our work differs from these works by modeling the absolute and relative temporal patterns between user behavior sequences and user interests in temporal recommendation.

7 CONCLUSION

In this paper, we aim to leverage the absolute time patterns and relative time patterns in user's behaviors sequence for sequential recommendation. A novel neural network framework named TASER are proposed, where the temporal patterns are captured by an absolute temporal module and a relative temporal module respectively, and a time-aware constraint is adopted into the loss function to obtain better representation of the temporal information. We conduct extensive experiments on seven datasets to verify the effectiveness of TASER on modeling temporal patterns. Experimental results show that TASER is able to achieve better performance noticeably, compared to the state-of-the-arts.

REFERENCES

[1] Ting Bai, Lixin Zou, Wayne Zhao, Pan Du, Weidong Liu, Jian-yun Nie, and Ji-Rong Wen. 2019. CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation. 675–684. https://doi.org/10.1145/3331184.3331199

[2] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Streaming recommender systems. In *Proceedings of the 26th WWW*. International World Wide Web Conferences Steering Committee, 381–389.

[3] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD*. ACM.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the Eleventh ACM Conference on Web Search and Data Mining*. ACM, 108–116.

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

[6] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In *Proceedings of the 14th ACM international CIKM*. ACM, 485–492.

[7] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation.. In *IJCAI*, Vol. 15. 2069–2075.

[8] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM RecSys*. ACM, 93–100.

[9] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2015. Content-Aware Point of Interest Recommendation on Location-Based Social Networks.. In *AAAI*.

[10] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* (2017).

[11] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM RecSys*. ACM, 131–138.

[12] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 191–200.

[13] 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.

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

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[16] George Karypis. 2001. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international CIKM*. ACM, 247–254.

[17] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD*. ACM, 447–456.

[18] 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 CIKM*. ACM, 1419–1428.

[19] Huizhi Liang, Yue Xu, Dian Tjondronegoro, and Peter Christen. 2012. Time-aware topic recommendation based on micro-blogs. In *Proceedings of the 21st ACM international CIKM*. ACM, 1657–1661.

[20] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003).

[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] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David MJ Tax. 2017. Interacting Attention-gated Recurrent Networks for Recommendation. In *Proceedings of the 2017 ACM on CIKM*. ACM, 1459–1468.

[23] 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.

[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. ACM.

[25] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive recurrent social recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 185–194.

[26] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.

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

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[30] Wenya Wang and Sinno Jialin Pan. 2018. Recursive Neural Structural Correspondence Network for Cross-domain Aspect and Opinion Co-Extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 2171–2181.

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

[32] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Richang Hong, Junping Du, and Meng Wang. 2017. Modeling the evolution of users' preferences and social links in social networking services. *TKDE* 29, 6 (2017), 1240–1253.

[33] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In *ACM RecSys*. 137–140.

[34] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD*. ACM, 723–732.

[35] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 211–222.

[36] Dawei Yin, Liangjie Hong, Zhenzhen Xue, and Brian D Davison. 2011. Temporal dynamics of user interests in tagging systems. In *Twenty-Fifth AAAI*.

[37] Dawei Yin, Liangjie Hong, Zhenzhen Xue, and Brian D. Davison. 2011. Temporal Dynamics of User Interests in Tagging Systems. In *Proceedings of the Twenty-Fifth AAAI (AAAI'11)*. AAAI Press, 1279–1285. http://dl.acm.org/citation.cfm?id=2900423.2900626

[38] 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*. ACM, 729–732.

[39] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. 2019. Next Item Recommendation with Self-Attentive Metric Learning. (2019).

[40] Yongfeng Zhang, Min Zhang, Yiqun Liu, Chua Tat-Seng, Yi Zhang, and Shaoping Ma. 2015. Task-based recommendation on a web-scale. In *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 827–836.

[41] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *Thirty-Second AAAI*.

[42] Meizi Zhou, Zhuoye Ding, Jiliang Tang, and Dawei Yin. 2018. Micro Behaviors: A New Perspective in E-commerce Recommender Systems. In *Proceedings of the Eleventh ACM Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 727–735. https://doi.org/10.1145/3159652.3159671

[43] 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 IJCAI-17*. 3602–3608.

[44] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 580–588.