Deep Behavior Tracing with Multi-level Temporality Preserved Embedding

Runze Wu[1], Hao Deng[1], Jianrong Tao[1], Changjie Fan[1], Qi Liu[2], Liang Chen[3]
[1]Fuxi AI Lab, NetEase Games, Hangzhou, China
[2]University of Science and Techonology of China, [3]Sun Yat-sen University
{wurunze1,denghao02,hztaojianrong,fanchangjie}@corp.netease.com,
qiliuql@ustc.edu.cn, chenliang6@mail.sysu.edu.cn

ABSTRACT

Behavior tracing or predicting is a key component in various application scenarios like online user modeling and ubiquitous computing, which significantly benefits the system design (e.g., resource pre-caching) and improves the user experience (e.g., personalized recommendation). Traditional behavior tracing methods like Markovian and sequential models take recent behaviors as input and infer the next move by using the most real-time information. However, these existing methods rarely comprehensively model the low-level *temporal irregularity* in the recent behavior sequence, i.e., the unevenly distributed time intervals between consecutive behaviors, and the high-level *periodicity* in the long-term activity cycle, i.e., the periodic behavior patterns of each user.

In this paper, we propose an intuitive and effective embedding method called **M**ulti-level **A**ligned **T**emporal **E**mbedding (MATE), which can tackle the temporal irregularity of recent behavior sequence and then align with the long-term periodicity in the activity cycle. Specifically, we combine time encoding and decoupled attention mechanism to build a temporal self-attentive sequential decoder to address the behavior-level temporal irregularity. To embed the activity cycle from the raw behavior sequence, we employ a novel temporal dense interpolation followed by a self-attentive sequential encoder. Then we first propose the periodic activity alignment to capture the long-term activity-level periodicity and construct the activity-behavior alignment to combine the activity-level with behavior-level representation to make the final prediction. We experimentally prove the effectiveness of the proposed model on a game player behavior sequence dataset and a real-world App usage trace dataset. Further, we deploy the proposed behavior tracing model into a game scene preloading service which can effectively reduce the waiting time of scene transfer by preloading the predicted game scene for each user.

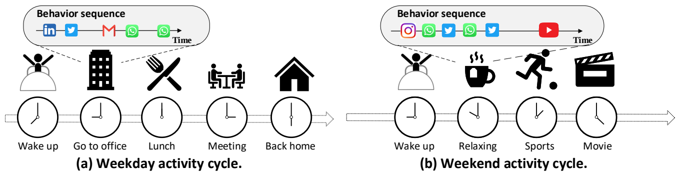

Figure 1: The example of weekday/weekend activity cycles and APP usage behavior sequence. In each subfigure, the arrow at the bottom denotes the time flow and each icon represents a specific activity; the arrow at the top is the time axis where each APP is a specific behavior and the position of each APP stands for the timestamp. We also observe the similar behavior/activity cycles in online games.

ACM Reference Format:
Runze Wu[1], Hao Deng[1], Jianrong Tao[1], Changjie Fan[1], Qi Liu[2], Liang Chen[3]. 2020. Deep Behavior Tracing with Multi-level Temporality Preserved Embedding. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412696

1 INTRODUCTION

Entering the age of big data and 5G, both online and offline systems like mobile phones and games have produced trillions of records of user behaviors (e.g., user id, timestamp, behavior type). For optimizing the system design (e.g, decrease the response latency in online games) and improving the user experience (e.g., personalized recommendation), user behavior modeling has become significantly important and drawn efforts from various domains [13, 15, 22].

One essential task in user behavior modeling is accurately tracing user behavior (e.g., predicting what a player will do next in online games), which indicates the ability to understand users' implicit activity. In this paper, we regard the observable action record of each user as a *behavior*, a sequence of behaviors with some specific intention as an *activity* and a sequence of activities in one day as a *cycle*, respectively. As shown in Fig. 1, user behavior sequence (APP usage records) can be segmented into different activities and the activity largely determines the scope of users' behavior (e.g., a user may check *social network* and *email* services in the activity of *Go to office*.) Recently, researchers have achieved great success in modeling users' behavior sequences in various online services like game behavior prediction [7], social networks service [27] and so on. Markov chain (MC) approaches [17] assume each behavior is dependent on the most recent history and map users and behaviors into latent spaces. Recurrent neural networks (RNN) models [5] relax the limitation of Markovian assumption and are more powerful in capturing complicated dependency. Self-attention networks (SAN) methods [30], enjoying the good parallelizability, show the superiority in sequential behavior modeling.

Despite the efforts made into behavior modeling, the multi-level temporality in behavior sequences is still largely under-explored. As shown in the bottom of Fig. 1, people tend to participate periodic, i.e., daily or weekly, activities (e.g., a weekday from waking up, going to office, lunch, meeting and going back home and a weekend which includes sports and entertainment) which we refer to as the *periodicity*. The existing methods for modeling behavior sequences usually follow specific assumptions on the periodicity [11], which limits the expressive power of modeling the complicated and implicit dependencies. On the other hand, the intervals between consecutive behaviors are not evenly distributed, which we refer to as the *temporal irregularity*. As shown in the top left of Fig. 1, the interval between *Twitter* and *Gmail* is much smaller than that between *Gmail* and *Whatsapp*. The same behavior sequence with different intervals can reveal discrepant activities or user statuses. The temporal irregularity has yet not been comprehensively addressed apart from bin encoding [30], specific gate mechanism [31], and simple concatenation [2]. The periodicity and temporal irregularity also limit the power of current methods to model game player behavior (see Fig. 3). Thus we lack an effective solution to address the multi-level temporality for behavior tracing.

In this paper, we first scrutinize the temporality of the online game player behavior sequence via an empirical analysis. Motivated by the observation, we propose **M**ulti-level **A**ligned **T**emporal **E**mbedding (MATE) model, which can effectively handle the temporal irregularity in behavior sequence and then align with the long-term periodicity in the activity cycle. To be specific, we address the behavior-level temporal irregularity by building a temporal self-attentive sequential decoder with time encoding and decoupled attention. For embedding the activity cycle from behavior sequence, we adopt a novel temporal dense interpolation and also construct a self-attentive sequential encoder. Then we first model the long-term periodicity with the periodic activity alignment and combine activity-level with behavior-level representation to predict the next behavior. To demonstrate the effectiveness of the proposed MATE, we conduct extensive experiments on a game player scene-transfer sequence dataset as well as the other real-world behavior sequence dataset. Furthermore, we deploy the proposed MATE into a real-time game scene preloading service which can preload the predicted game scene by effectively tracing the scene-transfer behavior of each player. The main contributions of this paper are outlined as follows:

- We comprehensively investigate the multi-level temporality in behavior sequences via empirical observation in a real-world online game.
- Behavior-level temporal irregularity and activity-level periodicity are captured by a novel hierarchical self-attention framework and the final prediction can be inferred via a delicate two-stage temporal alignment module.
- Extensive experiments on two real-world datasets verify the effectiveness of MATE, which is also verified by a deployed online game scene preloading service.

2 RELATED WORK

User Behavior Modeling. The research on behavior modeling can be mainly divided into two categories: activity recognition and behavior tracing. Conventional probability-based methods are applied to recognizing activity by modeling behaviors as categorical values directly [9]. Ryoo et al. implement Integral bag-of-words which can model the status of ongoing activities [18]. To trace behavior, Niebles et al. argue that it is critical to capture temporal context by dividing behavior sequence into motion segments [16]. Besides, Tao et al. construct an event2vec model that assigns each behavior a weight in terms of time gaps to target behavior [21]. There exist rare research efforts for modeling the long-term latent activity cycle consisting of a series of behaviors. In this paper, we focus on modeling the behavior-level and activity-level temporality for better modeling and tracing user behavior.
Temporality Modeling. How to encode or embed time information and effectively enhance the temporal representation is a vital step for sequential models, especially user behavior modeling. Apart from the indirect methods including bin encoding [30], interpolation [19] and simple concatenation [2], researchers also try to directly model continuous-time information. Zhu et al. propose a new time gate to enable the original recurrent neural network models to incorporate the behavior-level temporality [31]. Li et al. build a time-dependent representation in sequential models by two time embedding methods: time mask and event-time joint embedding [12]. Inspired by positional encoding in Transformer [24], recent researchers construct a trainable time embedding approach [8]. All the aforementioned temporal methods try to incorporate intervals between behaviors but rarely model the long-term activity-level periodicity. Some efforts have been made on the periodicity in the spatio-temporal forecasting [28, 32], which are not in the scope of this paper. In this research, we not only address the behavior-level temporality but also model activity-level periodicity.

3 PRELIMINARIES

Game scene. The game scene is the core element in online games where all kinds of players' behaviors happen, even some behaviors are restricted in a specific scene (e.g., attacking is forbidden in a peace-only scene). We collect a game player scene-transfer behavior sequence data for seven weeks (see detail in Sec. 5.1) from *JusticePC*[1], which is a popular massively multiplayer online role-playing game (MMORPG) released by NetEase Games[2] and creates a breathing virtual world. Various kinds of gameplays, along with the delicate game scenes with high definition, bring each player a feast of eyes as well as a high-end system requirement. The running game scene often occupies a lion's share of the resource of the client program of each player.

Usually, game clients do not load the resource of other game scenes until the game player leave the current scene as shown in Fig. 2(a). However, the waiting time for game scene transfer may be as long as tens of seconds, especially for main game scenes, which harms the experience of each player. How can we optimize the design of MMORPG to address this issue? A viable solution is to improve the client program or calibrate the game scene resource,

[1]https://leihuo.163.com/en/games.html?g=game-1#nsh
[2]https://www.neteasegames.com/

which is quite labor-intensive. Inspired by the cache mechanism, another possible idea is to intelligently trace scene-transfer behaviors of each player and preload the resource of the next game scene as shown in Fig. 2(b). To this end, we require an effective player scene-transfer behavior tracing model.

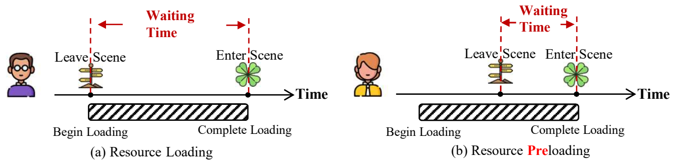

Figure 2: The two strategies of game scene resource loading.

Empirical Observation. Different from the traditional time-series tasks, there are two levels of significant temporality in the behavior sequence data. Fig. 3(a) shows that the time intervals between scene-transfer behaviors of each player are unevenly distributed and mainly follow a log-normal distribution. On the other hand, we investigate the long-term periodicity at a higher level, i.e., activity. Fig. 3(b) visualizes the behavior sequence during one week of a randomly sampled active player. We manually label the activity each behavior belongs to by consulting experienced game players so that each band represents a daily behavior sequence and the part in the same color denotes the same activity (e.g., daily request). We can observe that some activities are usually daily (or weekly) periodic. E.g., the sampled player usually first participates in "yellow" gameplays (joining a team) then in "orange" ones (daily request).

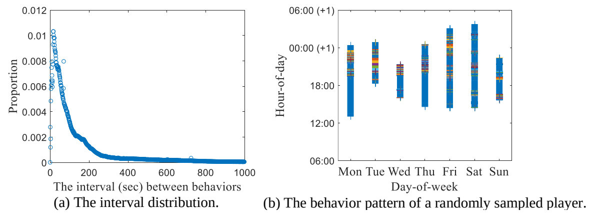

Figure 3: The empirical observation of the player behaviors.

Problem Definition. Given the behavior sequence as $\{(b_n, t_n)|n = 1, 2, \cdots\}$ where $b_n \in \{1, 2, \cdots, K\}$ is the behavior type, K is the number of the behavior types and $t_n \geq 0$ is the corresponding time. Given any j previous behaviors, our goal is to trace the $(j+1)$th behavior type based on the historical behavior sequence $\{(b_n, t_n)|n \leq j\}$ and the meta-data information for the prediction X^{meta} (e.g., the user profile, the temporal features such as hour-of-day and day-of-week). Formally, we propose MATE as a neural model f as follows:

$$\hat{p}_{j+1} = f(\{(b_n, t_n)|n \leq j\}, X^{\text{meta}}; \theta), \quad (1)$$
$$\hat{b}_{j+1} = \arg\max_{1 \leq k \leq K}\{\hat{p}_{j+1,k}\}.$$

θ is the model parameters. The model f infers the probability distribution of each type of the next behavior, $\hat{p}_{j+1}$ then easily obtains behavior prediction, $\hat{b}_{j+1}$. For better illustration, we summarize some important notations in Tab. 1.

Table 1: Some important notations.

Notation	Description	Shape
b_n	the type of the nth behavior	(1,)
t_n	the time of the nth behavior	(1,)
K	the number of the behavior types	(1,)
w	the size of the recent time window	(1,)
d	the number of hidden units	(1,)
X	the behavior type embedding vectors	(w,d)
Y	the implicit activity embedding vectors	(w,d)
H	the behavior sequence embedding	(w,d)
S	the activity sequence embedding	(w,d)
PE	the positional encoding vectors	(w,d)
TE	the time encoding vectors	(w,d)

4 PROPOSED MODEL

Here we introduce the basic sequential model and describe the enhanced MATE model as shown in Fig. 4.

4.1 Basic Sequence Modeling

In this subsection, we build a basic sequential model similar to Transformer-decoder [24] for user behavior tracing. The main idea is to embed the recent behavior sequence and combine the external meta-data information by building a non-linear model.

Behavior Type Embedding. We first project each behavior type b_n into a continuous embedding space as follows:

$$x_n = \text{One-Hot}(b_n)\mathbf{Z}_b. \quad (2)$$

Here we transform one-hot vectors into the behavior embedding vector $x_n \in \mathbb{R}^d$ by the shared behavior embedding matrix $\mathbf{Z}_b \in \mathbb{R}^{K\times d}$. d is the dimension of the embedding space.

Positional Encoding. We employ sinusoid encoding [24] as:

$$\text{PE}(\text{pos}, 2i) = \sin(\text{pos}/D^{2i/d}), \quad (3)$$
$$\text{PE}(\text{pos}, 2i+1) = \cos(\text{pos}/D^{2i/d}).$$

Here pos is the position and $2i, 2i+1$ are the dimensions. D is a large number, say 10000, which can fit a very long sequence.

Self-Attentive Sequential Representation. Considering the efficiency and effectiveness advantage in sequential modeling, we choose self-attention networks to model the behavior sequence. We set a time *window* to focus the most recent behavior sequence for tracing user behavior. Formally, we embed the w-most recent behaviors before position j with stacked N self-attention layers as[3]:

$$\begin{aligned} H^0 &= X + \text{PE}, \quad (4)\\ A^i &= \text{Attend}(H^{i-1}, H^{i-1}),\\ o^i &= \text{LayerNorm}(A^i + H^{i-1}),\\ H^i &= \text{LayerNorm}(o^i + \text{FFN}(o^i)). \end{aligned}$$

Here we refer X and PE as $\{x_n|j-w+1 \leq n \leq j\}$ and $\{\text{PE}_n|j-w+1 \leq n \leq j\}$, w is the size of the time window. We define Attend() as a scaled-product attention with query Q and key $\mathcal{K}$:

$$K, V, Q = W_k\mathcal{K}, W_v\mathcal{K}, W_q\mathcal{Q}, \quad (5)$$
$$\text{Attend}(\mathcal{Q}, \mathcal{K}) = \text{SoftMax}(\frac{Q^TK}{\sqrt{d}})V.$$

[3]For simplicity, we can denote the last three lines as $H^i = \text{ResAttend}(H^{i-1}, H^{i-1})$.

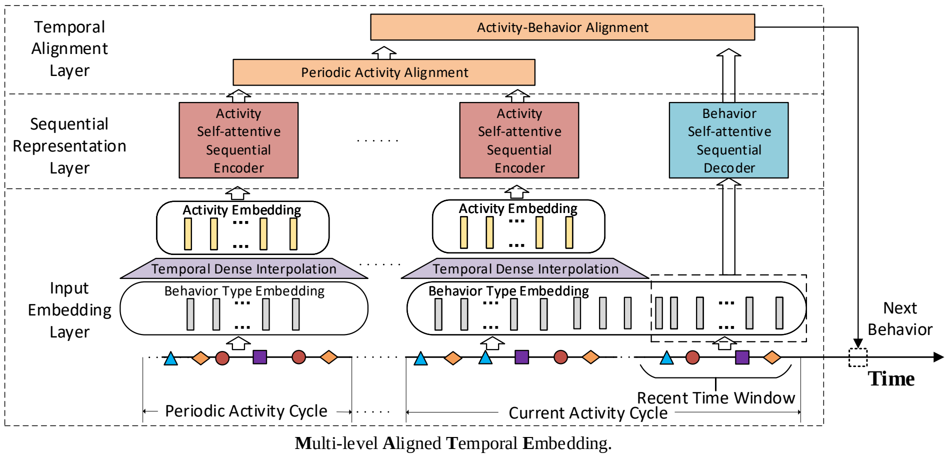
Temporal Alignment Layer
Activity-Behavior Alignment
Periodic Activity Alignment
Sequential Representation Layer
Activity Self-attentive Sequential Encoder
Activity Self-attentive Sequential Encoder
Behavior Self-attentive Sequential Decoder
Input Embedding Layer
Activity Embedding
Temporal Dense Interpolation
Behavior Type Embedding
Next Behavior
Time
Recent Time Window
Periodic Activity Cycle
Current Activity Cycle
Multi-level Aligned Temporal Embedding.

Figure 4: The framework of MATE model: the main idea is to align current *activity* with periodic ones (i.e., inferring the next activity according to the periodicity) then align the activity with the recent *behaviors* (i.e., inferring the next behavior given the inferred activity). The axis below denotes the time direction on which each shape represents a specific behavior and the dashed one is the target to predict.

Here we linearly map the inputting query and key by weight matrices W_k, W_v and W_q and d is the dimension of queries and keys. The position-wise feed-forward network (FFN) is a simple two-layer dense network as follows:

$$\text{FFN}(x) = \text{ELU}(xW_1 + b_1)W_2 + b_2. \tag{6}$$

Here we adopt exponential linear units (ELU) [3] as the activation function for the intermediate layer. We employ residual connection and layer normalization for model optimization. H^N is the implicit self-attentive sequence representation.

Meta Fusion. Given H^N, we can further infer the next behavior by incorporating the meta-data information. We concatenate the flattened behavior sequence representation and meta-data vector then employ a position-wise FFN for final prediction:

$$\begin{aligned} X_f &= \text{Concat}(\text{Flatten}(\text{LayerNorm}(H^N)), X^{\text{meta}}), \\ X_f' &= \text{LayerNorm}(\text{FFN}(X_f) + \text{Flatten}(H^N)), \\ \hat{p}_{j+1} &= \text{SoftMax}(W_f X_f'). \end{aligned} \tag{7}$$

Here $\hat{p}_{j+1}$ is the final probability distribution of the next behavior type and W_f is the final output weight matrix. We also utilize layer normalization and residual connection for better modeling.

Remark. The basic model can capture the interdependency between behavior sequence and fuse external meta-data information. However, there are two concerns not addressed. On one hand, the irregularly-spaced intervals between the recent behaviors are still not effectively explored. This behavior-level temporality can help distinguish behavior sequences with different intervals in the same order. On the other hand, the long-term impact of historical activity cycle is not taken into account. Simply adding the depending sequence length, say time window size w, does not necessarily improve the effectiveness of the sequential model (we verify that in Sec. 5.4) but dramatically complicate the model computation. As observed in Fig. 3(b), there exists some significant periodic patterns in the long-term activity cycle.

4.2 Behavior Self-Attentive Sequential Decoder

For the behavior-level temporality in the recent behavior sequence, i.e., temporal irregularity, we adopt time encoding and decoupled attention mechanism to construct behavior self-attentive sequential decoder as the right part of Fig. 4.

Time Encoding. We define time encoding as follows:

$$\begin{aligned} \tilde{t_n} &= \sum_{j=2}^{n} \text{Scale}(t_j - t_{j-1}), \\ \text{TE}(t_n, 2i) &= \sin(\tilde{t_n}/D^{2i/d}), \\ \text{TE}(t_n, 2i+1) &= \cos(\tilde{t_n}/D^{2i/d}). \end{aligned} \tag{8}$$

Here t_n and $\tilde{t}_n$ is the timestamp and scaled timestamp, $2i$ and $2i+1$ denote the dimension. The base number D is set to 20. For the first position, we set $\tilde{t}_1$ to 0. We pick $\log(x+1)$ to implement Scale(x) due to the observation that inter-behavior time mainly follows a log-normal distribution. The time encoding is equivalent to the positional encoding if the time intervals are evenly spaced.

Decoupled Attention. To further enhance the representation of temporal irregularity, we adopt the decoupled attention mechanism [4], denoted as DeAttend(), to refine Eq. (5) as follows:

$$\begin{aligned} K, V, Q &= W_k\mathcal{K}, W_v\mathcal{K}, W_q\mathcal{Q}, \\ R &= Q^T K + Q^T W_r \cdot \text{TE} \\ &\quad + \text{Tile}(u)^T K + \text{Tile}(v)^T W_r \cdot \text{TE}, \\ \text{DeAttend}(\mathcal{Q}, \mathcal{K}, \text{TE}) &= \text{SoftMax}(R/\sqrt{d})V. \end{aligned} \tag{9}$$

Note that we decouple the inputs of the attention module into two parts: X and TE, which are the behavior embedding and the time encoding respectively. We set $H^0 = X$ and factorize the attention module as the sum of four parts:

- $Q^T K$ denotes content-based addressing.
- $Q^T W_r \cdot \text{TE}$ represents a content-dependent temporal bias.
- $\text{Tile}(u)^T K$ encodes a sequence-level content bias.

- $\text{Tile}(v)^T W_r \cdot \text{TE}$ captures a sequence-level temporal bias.

$u \in \mathbb{R}^d$ and $v \in \mathbb{R}^d$ are two trainable vectors maintaining attentive bias of each query towards different keys. Tile() denotes a function which can tile a vector multiple times into a matrix.

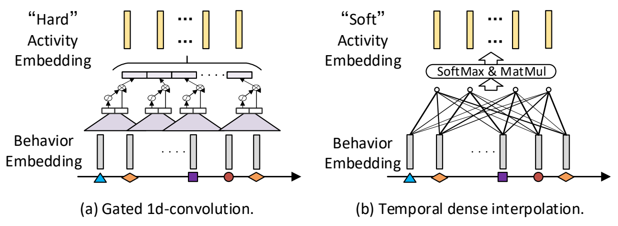
"Hard" Activity Embedding
Behavior Embedding
(a) Gated 1d-convolution.
"Soft" Activity Embedding
SoftMax & MatMul
Behavior Embedding
(b) Temporal dense interpolation.

Figure 5: The two methods for activity embedding.

4.3 Activity Self-Attentive Sequential Encoder

As the left and middle part of Fig. 4, we propose a novel activity embedding method and build an activity encoder.

Activity Embedding. As mentioned in the Introduction, we segment each day of behavior sequences into cycles of activities. However, the relationship between behavior and activity is usually implicit and unobservable unless time-consuming manually labeling. Considering the activity covering a sequence of behaviors, we can aggregate the consecutive behaviors to represent the implicit *activity* by *hard* segmentation methods such as gated 1d-convolution as shown in Fig. 5(a). Nevertheless, the intervals between behaviors are highly uneven (see Fig. 3(a)) so that simply aggregating consecutive behavior sequence is very not reasonable. Inspired by Trask et. al. [23], we propose a novel *temporal dense interpolation* as shown in Fig. 5(b) for *soft* segmentation. The pseudocode for a given sequence is shown below in Algorithm 1.

Algorithm 1 Temporal Dense Interpolation.

```
Input: the factor $F$, the behavior embeddings $X_1, X_2, \cdots, X_T$ and the
    relevant timestamps $t_1, t_2, \cdots, t_T$ (assuming the length of behavior
    sequence in the activity cycle is $T$).
Output: the activity embeddings $Y_1, Y_2, \cdots, Y_F$.
1: for $i = 1, 2, \ldots, F$ do
2:     for $j = 1, 2, \ldots, T$ do
3:         $\alpha_j = (1 - |(t_i - t_1)/(t_T - t_1) - j/(F+1)|)^2$
4:         $Y_i = Y_i + \alpha_j / Z_\alpha * X_j$
5:     end for
6: end for
7: return
```

Note that we compute the relative difference between each implicit activity and each behavior then aggregate the relevant embedding vectors with different weights, α_j. Besides, we adopt the normalization term Z_α to ensure the sum of weights α equal to 1. Algorithm 1 can be easily implemented by weight normalization and matrix multiplication. We assume that each cycle contains F activities and set $F = w$ consistent with recent time window.

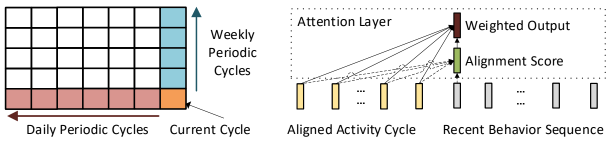

Figure 6: The temporal alignment: (a) Align the current cycle (the last cell) with previous daily (horizontal cells) and weekly (vertical cells) cycles; (b) Align each behavior (the grey bars) with the aligned activity cycle (the yellow bars).

Activity Encoder. Considering the embedded activities are evenly spaced, we follow the same procedure in Eq. (4) of the basic sequential modeling in Sec. 4.1:

$$S^0 = Y + \text{PE}, \tag{10}$$
$$S^i = \text{ResAttend}(S^{i-1}, S^{i-1}).$$

Here $S \in \mathbb{R}^{w \times d}$ denotes the activity sequence embedding. We also adopt M layers of self-attention module to encode the interdependency between activities. In this way, we can obtain activity representation S^M for each cycle.

4.4 Temporal Alignment

With the representation of historical/recent cycles of activities and the recent behaviors, we propose a two-step alignment module for final fusion as shown in Fig. 6 and the top of Fig. 4.

Periodic Activity Alignment. To effectively handle activity-level temporality, we first align the activity representation of the current cycle with periodic ones (e.g., one day or one week ago). We align the current activity with the periodic ones to capture the periodicity as follows:

$$\tilde{S} = \text{ResAttend}(S^M_{\text{current}}, S^M_{\text{periodic}}). \tag{11}$$

Here we take the current activity cycle as a query and the periodic ones as keys and values. For multiple periodic cycles, we concatenate them into one long activity cycle. Thus the aligned activity representation $\tilde{S}$ governs the activity-level periodicity.

Activity-Behavior Alignment. Next we align the current behavior sequence with the aligned activity cycle $\tilde{S}$ as follows:

$$\tilde{H} = \text{ResAttend}(H^N, \tilde{S}). \tag{12}$$

Here we use the recent behavior sequence representation as a query and the aligned activity cycle as keys and values. Therefore the aligned behavior sequence embedding $\tilde{H}$ not only decodes the current behavior-transfer dynamically but is aligned with the long-term activity-level periodicity. Further, we can combine $\tilde{H}$ with the external meta-data information X^{meta} to infer the next behavior type via similar steps in Eq. (7).

4.5 Implementation and Optimization

We implement the MATE model on an RTX-2080Ti equipped machine with 60GB memory using the TensorFlow library. The numbers of the stacked layers, N and M, and the size of the time window, w, are tuned by considering both efficiency and effectiveness. For better modeling, we use the multi-head mechanism [24] in Attend and DeAttend module. The loss is optimized by Adam algorithm [10] with a learning rate of 0.003. To reduce complexity, each activity encoder layer shares the same parameters. We also employ early stopping [25] and dropout [20] techniques to prevent over-fitting.

Table 2: The basic statistics of the datasets.

Data	#Behavior	Avg_daily_seq	Max_daily_seq	#Training	#Testing
MMORPG	810	29.5	246	8,483,736	4,117,168
App Usage	15.6	1,605	62	1,556,547	340,539

5 OFFLINE EXPERIMENTS

In this section, we experimentally compare our proposed MATE[4] against other baseline methods on two datasets including MMORPG scene-transfer behaviors and real-world App usage behaviors.

5.1 Datasets

We briefly show the statistics of the two offline datasets in Tab. 2.
MMORPG Scene-Transfer. The scene-transfer behavior sequence dataset is collected from one game server of *JusticePC* between July 1 and August 18, 2019. There are more than 58,000 players and over 50 million scene-transfers across nearly 2,000 game scenes. Considering many players stay online very late till two or three AM of the next day (see Fig. 3(b)), we segment the raw sequence to cycles, i.e., 0600 AM to 0600 AM. We merge the consecutive records of the same scene and eliminate the cycles with less than 10 records and the players with less than 50 records and focus on the most frequent 810 scenes. Finally, we take the 14-day records between July 29 and August 11 as the training set and that between August 12 and 18 as the testing set so that each record can be related to the recent 6-day daily and 4-week weekly sessions. The external meta-data features contain level, role class, hour-of-day, and day-of-week.
TsingHua App Usage Trace[5]. [26] This dataset collects the usage activity of 1,000 users of one city in China for one week (June 20 to 26, 2016). Each entry contains the anonymized user ID, timestamps and App ID. We segment the raw sequence of each day into different cycles. We first merge the repeating records of the same app then eliminate the sessions with less than 5 records and the players with less than 10 records and focus on the top 1,605 apps dominating 99.99% of all the usage records. Limited by the time span, we take sessions between the 2nd to 6th day as the training set and that on the last day as the testing set so that each record can be related to the one-day-ago sessions. We take hour-of-day as the external meta-data feature.

Table 3: The parameter setting for MATE.

Data	d	w	M	N	max_cycle_len	#attention_head
MMORPG	32	10	1	2	256	8
App Usage	32	5	1	1	64	4

[4]Code available: https://github.com/fuxiAIlab/MATE
[5]http://fi.ee.tsinghua.edu.cn/appusage/

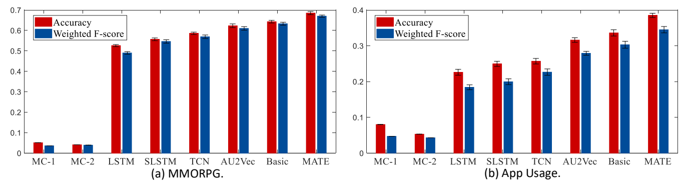

Figure 7: The behavior tracing performance comparison.

5.2 Settings

Baselines. We choose the following baseline methods:

- **Markov Chain:** We adopt Markov chain with the order of one and two denoted as **MC-1** and **MC-2**, respectively.
- **LSTM:** [6] LSTM is employed to model the nonlinear dependency over the sequence to predict the next behavior.
- **Soft-attention LSTM:** [14] An extended LSTM for capturing long-range dependency denoted as **SLSTM**.
- **Temporal Convolutional Network:** [1] A sequential model can obtain a larger receptive field with hierarchical convolutional layers denoted as **TCN**.
- **AppUsage2Vec:** [29] Denoted as **AU2Vec**, this model adopts an attention-based model to embeds multiple types of features for behavior prediction.
- **Basic:** We adopt a Transformer-decoder [24] based model to infer the next behavior given the recent w behaviors.

Parameter settings. We set the parameters of MATE for two datasets as shown in Tab. 3. [6] The length of the input sequences of LSTM, SLSTM, TCN, and AU2Vec is the same as MATE while Basic takes the recent w behaviors as inputs due to the high complexity. We additionally compare **Basic(2d)**/**Basic(1w)** given the behavior sequences in the recent two days/one week as input in Sec. 5.4.
Metrics. We adopt *Accuracy* and *Weighted F-score* to show the performance for behavior tracing task. The higher the metric is, the better the model performs.

5.3 Behavior Tracing Performance

Fig. 7 shows the overall performance of different methods on two datasets. The results in both scenarios prove that our proposed MATE yields the best performance. SAN-based models, i.e., Basic and MATE, are very superior to the traditional methods including MC-1, MC-2, and LSTM. Non-parametric Markov models perform poor possibly because MC in lower order is hard to capture interdependency of long sequences. LSTM, SLSTM and TCN gain significant improvements by capturing long sequential dependency. AppUsage2Vec performs quite well by using attention mechanisms to incorporate rich external features. Our proposed MATE, which outperforms the other baselines, not only takes advantage of the superiority of sequential modeling of SAN methods but also effectively combines the rich temporal features including the behavior-level and activity-level temporality. Specifically, MATE gains 5.3%, 8.7% improvements in terms of accuracy of MMORPG than the most competitive baselines Basic and AU2Vec and achieves 14.6%, 21.8% improvements in terms of accuracy of App Usage. We also conduct a paired t-test and conclude that our proposed MATE is significantly

[6]The parameter sensitivity is not given due to the limited space.

better than the other baselines with the p-value less than 0.02 in both datasets.

Table 4: Ablation study of the proposed model.

Method	Accuracy (MMORPG)	Accuracy (App Usage)
Basic	0.643	0.336
Basic+TE	0.661	0.357
MATE Decoder	0.665	0.359
MATE (Only current activity cycle)	0.671	0.376
Time-LSTM [31]	0.532	0.246
Basic+time2vec [8]	0.659	0.358
Basic+time_mask [12]	0.647	0.345
Basic+event_time_joint [12]	0.648	0.346
Basic(2d)	0.643	0.335
MATE(2d)	0.671	**0.385**
Basic(1w)	0.641	N/A
MATE(1w)	0.674	N/A
MATE(gated_conv1d)	0.677	0.375
MATE	**0.687**	**0.385**

5.4 Ablation Study

We test the effects of different components, i.e., behavior decoder, current and periodic activity encoder, of the proposed MATE. We also compare the performance of our model with different settings such as the length of inputs, the time encoding method and the activity embedding approach.

Tab. 4 shows that both time encoding and decoupled attention boost the performance by capturing the behavior-level temporality. And activity encoders effectively enhance the proposed model by learning the activity-level periodicity. On the other hand, LSTM-based temporal embedding methods [31] are not as effective as SAN-based ones but gain improvements compared with the original LSTM. Our proposed time encoding performs similarly to time2vec but outperforms other time embedding methods like time mask and event-time joint embedding. Compared with the basic sequential modeling, MATE captures the long-term periodicity and gains the advantages even given the same-length input sequence (two days or one week). For activity embedding methods, our proposed temporal dense interpolation performs better than "hard" gated-conv1d.

5.5 Case Study

To verify the Periodic Activity Alignment in Sec. 4.4, we randomly sample a behavior sequence of a game player in MMORPG[7] dataset and manually label the activities of the last three days as shown in Tab. 5. The red *daily request* is the last activity covering the target behavior to predict. Fig. 8 shows that *join team* $\longrightarrow$ *daily request* is a daily periodic activity-transfer so that these two activities draw the most attention from *join team* of Day 3. In addition, *trading* is kind of related to *check item*, and *wandering* is nearly random and not dependent on previous activities. Thus our MATE can align the current activity with periodic activity cycle for behavior tracing.

6 ONLINE APPLICATION

Verified with the offline datasets from MMORPG and App Usage, the MATE model is also deployed in *JusticePC* of NetEase Games. In this section, we introduce the online game scene preloading service, which can reduce the waiting time of scene transfer for each game player by preloading the next game scene predicted by our proposed MATE. As shown in Fig. 9, the whole service includes three modules, i.e., data, training and serving module.

[7] App Usage dataset isn't used due to no information about each APP except ID.

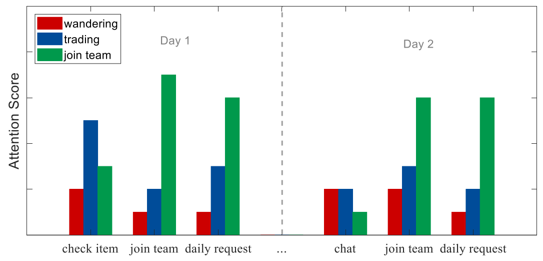

Figure 8: The attention score of activities of Day 3 (the colored bars) on activities of last two days (the x-axis).

Table 5: The labeled activity cycles of three days.

Day	Labeled Activity Cycle
1	*Check Item* $\longrightarrow$ *Join Team* $\longrightarrow$ *Daily Request* $\longrightarrow \cdots$
2	*Chat* $\longrightarrow$ *Join Team* $\longrightarrow$ *Daily Request* $\longrightarrow \cdots$
3	*Wandering* $\longrightarrow$ *Trading* $\longrightarrow$ *Join Team* $\longrightarrow$ *Daily Request*

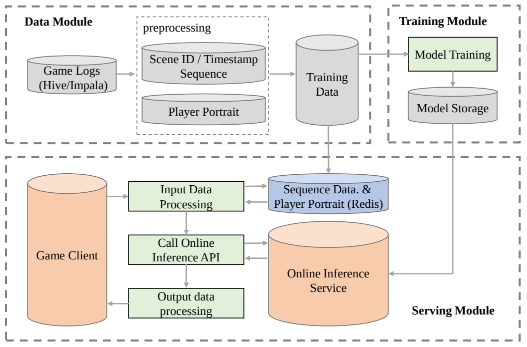

Figure 9: The online game scene preloading service.

6.1 Data Module

The data module is made up of two steps, i.e., data storage and preprocessing. We use Hive and Impala as data storage facilities, both of which are popular data warehouse softwares for managing large-scale datasets residing in distributed storage using SQL. On each day, we store the game scene-transfer behaviors (including the ID of each scene and the corresponding timestamp) of the last day as sequences and preprocess the updated player portrait and temporal features as the external meta data. In this way, we prepare the training data source for learning the parameters of MATE model.

6.2 Training Module

We implement the offline training module by TensorFlow using the tuned hyper-parameters. For each behavior sequence, we adopt a method of sliding window to obtain the "current cycle" and take the next scene as the target. According to the timestamp of each behavior, we can get the relevant periodic cycles (the non-existing cycles can be set to None). When training done, we save the model parameters with the best performance. The saved model can be deployed to an online inference service, in which TensorFlow Serving can help perform flexible and efficient machine learning inference.

6.3 Serving Module

To realize a real-time preloading service, we devise a robust online serving architecture. The online serving module contains three services: processing service, storage service and inference service. When entering a new game scene, the client of each player sends a request conveying the ID of the scene and the timestamp to our processing service. Receiving the request, our processing service combines the historical behavior sequences to obtain the input data of the MATE model as well as saves the new scene-transfer behavior into the storage serve. The storage service is implemented by a Redis database, which also stores the player portrait features and the periodic cycles. Given the input data, our processing service calls online inference service by an API to obtain the probabilities of each class of the next scene. At last, the processing service replies to the game client with the most possible ID of the game scene. In practice, we also filter the output results unless the probability of the most possible scene is above our preset threshold. A/B testing shows that the deployed game scene preloading service reduces nearly **30%** of the average waiting time for scene transfer.

7 CONCLUSION

In this paper, we proposed a novel model MATE to trace user behavior. Based on the basic sequential model, we managed to capture the significant behavior-level and activity-level temporality. We built a behavior decoder by combining time encoding and decoupled attention to handle temporal irregularity of behavior sequence. For implicit activity, we first embedded activity by aggregation, which was fed into a stacked attention-based activity encoder. We temporally aligned the current activity cycle with the previous periodic ones to incorporate the periodicity into activity inferring. Further, we aligned the behavior decoder with the aligned activity to make the final prediction. We verified the effectiveness of MATE on a game player scene-transfer dataset and an App usage dataset. An online game scene preloading service equipped with MATE also significantly reduces the waiting time for game scene transfer to improve the experience of each player.

REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[2] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 6085.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* abs/1511.07289 (2015).

[4] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv* abs/1901.02860 (2019).

[5] 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.

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

[7] Adam Katona, Ryan J. Spick, Victoria J. Hodge, Simon Demediuk, Florian Block, Anders Drachen, and James Alfred Walker. 2019. Time to Die: Death Prediction in Dota 2 using Deep Learning. *ArXiv* abs/1906.03939 (2019).

[8] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2Vec: Learning a Vector Representation of Time. *arXiv* abs/1907.05321 (2019).

[9] Eunju Kim, Sumi Helal, and Diane Cook. 2009. Human activity recognition and pattern discovery. *IEEE pervasive computing* 9, 1 (2009), 48–53.

[10] Diederik P Kingma and Jimmy Ba. [n.d.]. Adam: A method for stochastic optimization. *arXiv* abs/1412.6980 ([n. d.]).

[11] Takeshi Kurashima, Tim Althoff, and Jure Leskovec. 2018. Modeling interdependent and periodic real-world action sequences. In *Proceedings of the 2018 World Wide Web Conference*. 803–812.

[12] Yang Li, Nan Du, and Samy Bengio. 2017. Time-dependent representation for neural event sequence prediction. *ArXiv* abs/1708.00065 (2017).

[13] Hongyu Lu, Min Zhang, Weizhi Ma, Ce Wang, Feng xia, Yiqun Liu, Leyu Lin, and Shaoping Ma. 2019. Effects of User Negative Experience in Mobile News Streaming. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 705–714.

[14] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[15] Matthew Mitsui and Chirag Shah. 2019. Bridging gaps: Predicting user and task characteristics from partial user information. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 415–424.

[16] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. 2010. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*. Springer, 392–405.

[17] 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.

[18] Michael S Ryoo. 2011. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *2011 International Conference on Computer Vision*. IEEE, 1036–1043.

[19] Myron Scholes and Joseph Williams. 1977. Estimating betas from nonsynchronous data. *Journal of financial economics* 5, 3 (1977), 309–327.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[21] Jianrong Tao, Jiarong Xu, Linxia Gong, Yifu Li, Changjie Fan, and Zhou Zhao. 2018. NGUARD: A Game Bot Detection Framework for NetEase MMORPGs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 811–820.

[22] Paul J Taylor, Darlene F Russ-Eft, and Daniel WL Chan. 2005. A meta-analytic review of behavior modeling training. *Journal of applied psychology* 90, 4 (2005), 692.

[23] Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. *arXiv* abs/1506.02338 (2015).

[24] 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.

[25] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation* 26, 2 (2007), 289–315.

[26] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 174.

[27] Linyun Yu, Peng Cui, Fei Wang, Chaoming Song, and Shiqiang Yang. 2015. From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics. In *2015 IEEE International Conference on Data Mining*. IEEE, 559–568.

[28] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[29] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. 2019. AppUsage2Vec: Modeling Smartphone App Usage for Prediction. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1322–1333.

[30] 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 Conference on Artificial Intelligence*.

[31] 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 *IJCAI*. 3602–3608.

[32] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. 2018. Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns.. In *IJCAI*. 3732–3738.