# DPGN: Denoising Periodic Graph Network for Life Service Recommendation

Hao Xu*
Meituan
Beijing, China
kingsleyhsu1@gmail.com

Huixuan Chi*‡
Institute of Computing Technology,
Chinese Academy of Sciences
Beijing, China
chihuixuan21s@ict.ac.cn

Danyang Liu
Meituan
Beijing, China
danyangliucs@gmail.com

Sheng Zhou
Zhejiang University
Hangzhou, China
zhousheng_zju@zju.edu.cn

Mengdi Zhang†
Meituan
Beijing, China
mdzhangmd@gmail.com

## ABSTRACT

Different from traditional e-commerce platforms, life service recommender systems provide hundreds of millions of users with daily necessities services such as nearby food ordering. In this scenario, users have *instant intentions* and *living habits*, which exhibit a *periodic* tendency to click or buy products with similar intentions. This can be summarized as the *intentional periodicity* problem, which was not well-studied in previous works. Existing periodic-related recommenders exploit time-sensitive functions to capture the evolution of user preferences. However, these methods are easily affected by the real *noisy signal* in life service platforms, wherein the recent noisy signals can mislead the instant intention and living habits modeling. We summarize it as the *noise* issue. Although there are some denoising recommenders, these methods cannot effectively solve the noise issue for intentional periodicity modeling.

To alleviate the issues, we propose a novel **D**enoising **P**eriodic **G**raph **N**etwork (DPGN) for life service recommendation. **First**, to alleviate the noisy signals and model the instant intention accurately, we propose (i) temporal pooling (TP) to encode the most representative information shared by recent behaviors; (ii) temporal encoding (TE) to encode the relative time intervals. **Second**, to capture the user's living habits accurately, we propose the *memory mechanism* to maintain a series of instant intentions in different time periods. **Third**, to further capture the intentional periodicity, we propose the *temporal graph transformer* (TGT) layer to aggregate temporal information. **Last**, the *denoising task* is further proposed to alleviate the noisy signals. Extensive experiments on both real-world public and industrial datasets validate the state-of-the-art performance of DPGN. Code is available in https://github.com/ytchx1999/DPGN.

* Both authors contributed equally. ‡ Work done during his internship at Meituan.
† Corresponding author.

## CCS CONCEPTS

• **Information systems → Recommender systems**;

## KEYWORDS

Intentional Periodicity, Denoising, Life Service Recommendation

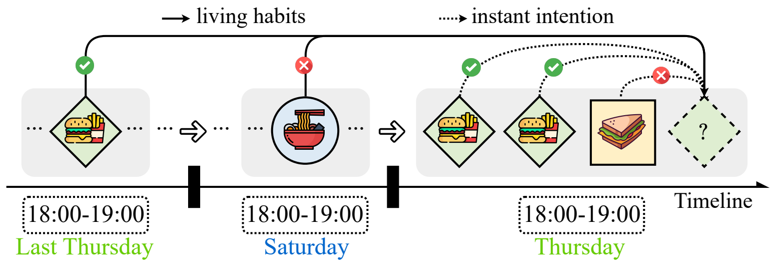
**ACM Reference Format:**
Hao Xu*, Huixuan Chi*‡, Danyang Liu, Sheng Zhou, and Mengdi Zhang†. 2023. DPGN: Denoising Periodic Graph Network for Life Service Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614850

## 1 INTRODUCTION

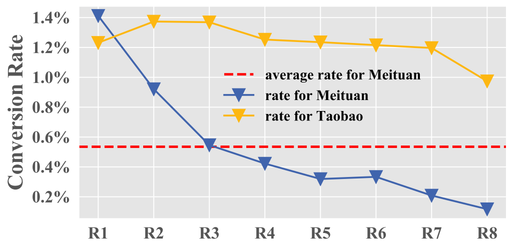
Large-scale life service platforms such as Alibaba Life Service[1], Meituan[2], and Uber Eats[3] provide services for hundreds of millions of users. Different from traditional e-commerce platforms [45, 46], life service platforms recommend daily necessities such as nearby delicacies to users at a specific time, which provides users with a more seamless life experience. Compared with other types of recommender systems, life service recommendation has the following characteristics/challenges: (1) *instant intention.* Users have real-time needs and often click or buy items with specific intentions. The "intention" reveals the high-level semantics of items (e.g. category). For instance, users express interest in fast food during lunchtime and prefer to collect their meals within an hour of purchase. (2) *living habits.* Users lead a regular lifestyle and exhibit a tendency to click or purchase items *periodically* with similar intentions. For instance, a user will buy burgers every Thursday night and prefer to order noodles every Saturday night. These characteristics have important implications for life service recommendation but have rarely been studied by previous works. We summarize it as an *intentional periodicity* problem. In this paper, we focus on how to model such intentional periodicity in life service platforms.

Let us first review existing periodic-related recommenders. Earlier methods [7, 30, 41] adopt the self-attention mechanism into historical long sequences. Although these methods capture the dynamics of user preferences, they are not time-aware and do not

[1]https://www.ele.me/
[2]http://i.meituan.com/
[3]https://www.ubereats.com/



(a) The motivation example in real-world life service platforms.



(b) Purchase conversion rate on different datasets.

**Figure 1: The analysis for the impact of noisy signals on intentional periodicity modeling. (a) is the motivation example in life service platforms. As for noisy signals, (b) is the statistic of the purchase conversion rate on Meituan and Taobao datasets. The groups labeled as "Rx" on the x-axis are divided by the category's ranking, where the ranking is based on the click number in sub-sequences. For instance, "R1" denotes categories with the No.1 click number in each sub-sequences.**

explicitly encode the periodicity. Recent works [4, 13, 28, 32, 36] capture the evolution or periodicity of user intentions by utilizing time interval contexts and time-decaying functions. However, these methods still fail to capture the intentional periodicity in life service platforms. *First*, since they are sensitive to time and focus more on recent interactions, the recent real-world *noisy signals* will lead to unreliable instant intention modeling. *Second*, the living habits, which is composed of a series of instant intentions, are also affected by noisy signals. We summarize it as the *noise* issue. In short, the above existing solutions fail to address the noise issue for intentional periodicity modeling.

To further investigate the noise issue for intentional periodicity modeling, we first define the *noisy signals* as interactions (e.g. clicks) that cannot align with the user's real intentions in life service platforms. For instance, in Figure 1(a), since the user's instant intention between 18:00-19:00 is to order a Burger package, the most recently clicked Sandwich is a noisy signal. These noisy signals can mislead the model and result in recommending another Sandwich instead of the desired Burger package. Furthermore, to illustrate the presence of noisy signals in instant intentions, we made statistics of the purchase conversion rate compared with different types of scenarios in Figure 1(b). The blue line (Meituan dataset) shows a rapid decrease with the decrease of click numbers, while the yellow line (Taobao dataset [47]) shows no significant changes. This indicates that users in life service platforms exhibit instant intentions, which can be reflected by their click behaviors within a short period (e.g., a user's sub-sequence within 1 hour). Besides, we observe that categories with low click numbers (in low ranking groups after R4) have below-average purchase conversion rates on Meituan dataset. This validates that click behaviors, which are often affected by various factors such as clickbait [24], contain noisy signals in life service platforms.

Although there are some denoising recommenders, these methods cannot effectively resolve the noise issue in life service platforms. First, most of these works [5, 33, 37, 39] define noisy signals based on implicit feedback, where false-positive interactions are identified using the user's rating score. However, they are not suitable for life service platforms. In these platforms, a low rating score is more likely to reflect the poor quality of items rather than indicating the user's dislike for a particular category. Second, most of these works overlook the crucial intentional periodicity modeling. These methods lack time-awareness and are incapable of effectively capturing the periodicity of user intentions.

To this end, we aim to alleviate noisy signals and model intentional periodicity with a combination of two factors: *instant intention* and *living habits*. In this work, we propose DPGN, a novel **D**enoising **P**eriodic **G**raph **N**etwork for life service recommendation. **First**, to alleviate the noisy signals and model the instant intention accurately, we propose two functions: (i) *temporal pooling* (TP), which encodes the most representative information shared by recent behaviors within a temporal window size to replace the last behavior; (ii) *temporal encoding* (TE), which explicitly encode the relative time intervals. **Second**, to capture the users' living habits, we propose the *memory mechanism*. Building upon the accuracy of instant intentions achieved through TP and TE, the memory mechanism maintains a series of instant intentions across different time periods. **Third**, to further capture the intentional periodicity, the *temporal graph transformer* (TGT) layer is proposed to aggregate temporal information from neighbors. **Last**, the multi-task training with *denoising task* is further proposed to alleviate the noisy signals. The contributions of the paper are summarized as:

- We take the first step to investigate the noise issue and intentional periodicity problem in real-world life service platforms.
- We propose a novel DPGN method for life service recommendation to tackle the issues. (1) To alleviate the noisy signals and model the instant intention accurately, we propose the temporal pooling (TP) and temporal encoding (TE) functions. (2) To capture the users' living habits, we propose the memory mechanism. (3) To further capture the intentional periodicity, we propose the temporal graph transformer (TGT) layer. (4) The multi-task training with denoising task is further proposed to alleviate the noisy signals.
- We conduct extensive experiments on both real-world public and industrial datasets. Experimental results verify the proposed method consistently brings remarkable improvements to previous state-of-the-art methods.

## 2 RELATED WORKS

### 2.1 Periodic-related Recommenders

Temporal dynamics is an important factor in user preference modeling. Earlier works [7, 30, 31, 41] attempt to capture dynamic user preference using sequential methods such as RNN [22] and self-attention [35] from historical long sequences. For instance, GRU4Rec [41] applies RNN to infer users' future intentions based on their historical click behaviors. SASRec [7] applies self-attention networks to model user behavior sequences, leading to improved recommendation performance. Despite the remarkable success, these methods overlook the temporal information in modeling historical interactions. More recently, several works [13, 14, 44] have focused on leveraging temporal information for user preference modeling. For instance, TiSASRec [13] utilizes the time intervals between interactions to improve the self-attention mechanism. TIEN [44] utilizes the rich interaction information in the time-aware item behavior for CTR prediction. Despite their success, these methods do not simultaneously consider temporal information, long-term preference, and short-term preference, all of which are crucial for modeling periodicity. To this end, some works [2, 4, 8, 12, 16, 20, 26, 28, 32, 36] explore different utilization of time-decaying functions to consider both the long-term and short-term preferences. For instance, NARM [12] and STAMP [16] employ a global encoder to capture general intentions, and a local encoder to model short-term intentions. TGSRec [4] and TGN [28] utilize a generic time encoding function to explicitly encode the periodicity. SLRC [36] proposes a Hawkes process into collaborative filtering for repeat consumption. However, as mentioned before, most of these works overlook the presence of real-world noisy signals in life service platforms, which can greatly affect the intentional periodicity modeling.

### 2.2 Denoising Recommenders

Recently, significant attention has been dedicated to the robustness of recommender systems, since existing recommendation approaches are vulnerable to various noisy signals. One line of research [5, 33, 37, 39] aims to denoise implicit feedback through different training strategies. For instance, ADT [37] proposes an adaptive denoising training by discarding the large-loss samples or lowering the weight of large-loss samples. SGDL [5] applies self-guided learning to divide the training phase into a noise-resistant period and a noise-sensitive period. RGCF [33] proposes a graph-denoising module and a diversity-preserving module for GNN-based recommenders. Another line of research [15, 18, 29, 40, 42] does not explicitly denoise implicit feedback but aims to improve the robustness of the model to alleviate noisy signals. For instance, Mult-VAE [15] applies a variational auto-encoder to build more robust recommender systems. SGL [40] adopts graph augmentation and contrastive learning to denoise for graph collaborative filtering. CL4SRec [42] proposes three sequential data augmentation approaches to construct self-supervision signals. However, as mentioned in Introduction, most existing recommenders define noisy signals from the perspective of implicit feedback, which is not suitable for life service platforms. Moreover, these denoising methods ignore the important periodicity modeling.

## 3 PRELIMINARY

In this section, we present fundamental notations and definitions. We begin by introducing the temporal user-item graph, which serves to record all user-item interactions as follows:

Definition 1. ***Temporal User-Item Graph*** *is defined as $\mathcal{G} = \{\mathcal{U}, \mathcal{I}, \mathcal{E}\}$, where $\mathcal{U}$ and $\mathcal{I}$ is a set of users and items, $\mathcal{E}$ is a set of temporal edges. Each edge $e^t_{u,i} \in \mathcal{E}$ is denoted as a triple, $e^t_{u,i} = (u, i, t)$, where $u \in \mathcal{U}$, $i \in \mathcal{I}$, and $t \in \mathbb{R}^+$ represents the timestamp. Each $(u, i, t)$ means that user $u$ has an interaction with item $i$ at time $t$. We denote $\mathcal{N}^t_u = \{(i, t_i) \mid e^{t_i}_{u,i} \in \mathcal{E}, t_i < t\}$ as the temporal neighborhood set of user $u$ in time interval $[0, t)$.*

This paper focuses on the life service recommendation problem in a continuous time setting. We then define the life service recommendation task as follows:

Definition 2. ***Life Service Recommendation Task.*** *Given a temporal user-item graph $\mathcal{G}$ and a new target $(u, t)$, the life service recommendation task aims to recommend an item list from $\mathcal{I}$ that user $u$ would be interested in at time $t$.*

## 4 METHODOLOGY

To address the aforementioned challenges, we introduce our proposed **D**enoising **P**eriodic **G**raph **N**etwork (DPGN) for life service recommendation, which consists of the following parts: (1) **Instant Intention Modeling**, which alleviates the noisy signals and models the instant intention more accurately by the proposed *temporal pooling* and *temporal encoding* functions. And these functions will be utilized multiple times in subsequent components. (2) **Memory Mechanism of Living Habits**, which captures the users' living habits and memorizes a series of accurate instant intentions by the proposed *memory mechanism.* (3) **Temporal Graph Transformer Layer**, which further captures the intentional periodicity. (4) **Multi-task Training**, which proposes a novel *denoising task* to alleviate the noisy signal. Figure 2 illustrates our proposed DPGN method.

### 4.1 Instant Intention Modeling

In this section, we describe two essential functions for instant intention modeling: (1) *temporal pooling*, which alleviates the noisy signals for instant intentions; (2) *temporal encoding*, which explicitly encodes the relative time intervals. Note that these functions will be utilized multiple times in subsequent components.

#### 4.1.1 *Temporal Pooling (TP).*

Since the new interaction at time $t$ could be a noisy signal, the calculation at time $t$ is incapable to represent the real instant intention. To alleviate the noisy signals for instant intentions, we design a temporal pooling function based on our observation discussed in Introduction. The intuition is to *encode the most representative information shared by recent behaviors within a temporal window size, replacing the last behavior.* The temporal pooling function $\psi(\cdot) \to \mathbb{R}^d$ to replace embedding $\boldsymbol{x}^t \in \mathbb{R}^d$ (affected by noisy signals) can be written as:

$$\psi\left([\boldsymbol{x}^{t_1}, \cdots, \boldsymbol{x}^t], \epsilon\right) = \mathrm{Mean}([\underbrace{\boldsymbol{x}^{t_1}, \cdots,}_{\text{dropout}} \underbrace{\boldsymbol{x}^{t_j}, \cdots, \boldsymbol{x}^t}_{\epsilon\text{-recent}}]), \tag{1}$$

where $t_1 \leq \cdots \leq t$ and $[\boldsymbol{x}^{t_1}, \cdots, \boldsymbol{x}^t]$ is a time-ordered list of embeddings. $\epsilon$ is a hyper-parameter to control the number of the

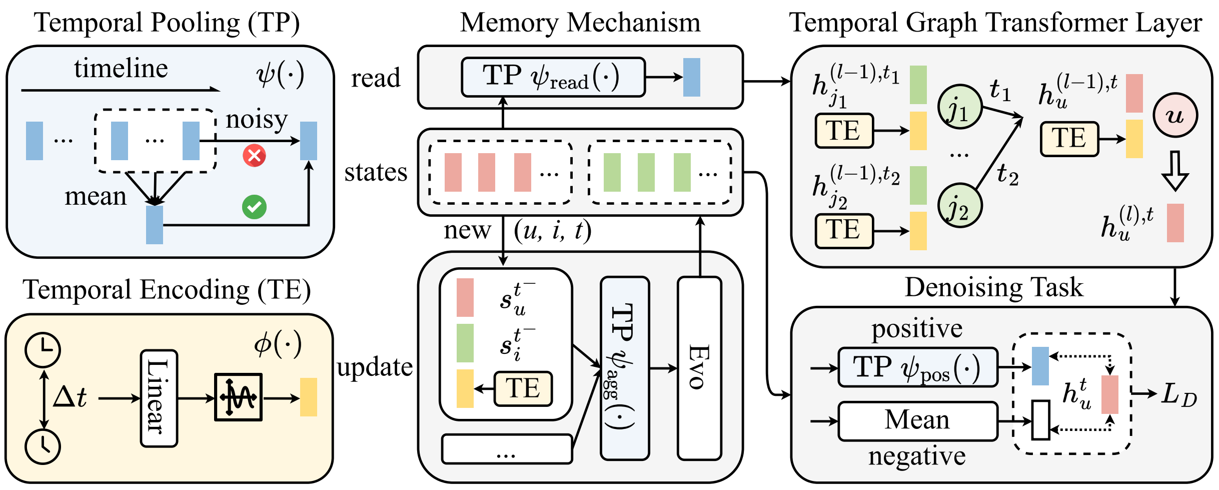


**Figure 2: The overall framework of our DPGN. We take the calculation of the hidden embedding $h_u^{(l),t}$ for user $u$ at time $t$ as an example (item follows the same process). (1) To model the instant intention more accurately, the temporal pooling (TP) $\psi(\cdot)$ is employed to alleviate the noisy signals and the temporal encoding (TE) $\phi(\cdot)$ is to encode the time intervals. (2) To capture the users' living habits, the memory maintains the states for users and items. The memory states are updated with new interactions and are smoothly read out. (3) To further capture the intentional periodicity, the temporal graph transformer (TGT) layer combines the hidden embedding and temporal encoding, and aggregates this information from neighbors to obtain the output embedding $h_u^{(l),t}$. (4) To further alleviate the noisy signals, the denoising task constructs new positive and negative samples.**

temporal window size. The temporal window size selects the most recent $\epsilon$ embeddings until time $t$ and the mean operation Mean$(\cdot)$ is used to encode the most representative information. Note that other operations including summation are possible. The output of $\psi(\cdot)$ can be used to replace $\boldsymbol{x}^t$ to alleviate the noisy signals.

*4.1.2* ***Temporal Encoding (TE).*** Motivated by recent works [4, 43], the intentional periodicity can be reflected by the time intervals between the user's different interactions. We use a generic time encoding function as $\phi(\cdot) \to \mathbb{R}^{d_t}$ based on Bochner's Theorem [19] to explicitly encode the time intervals into embedding. Specifically, given two interactions of the same user $(u, i_1, t_1)$ and $(u, i_2, t_2)$, and the time interval is defined as $\Delta t = |t_2 - t_1|$, we implement $\phi(\cdot)$ as:

$$\phi(\Delta t) = \left[\cos(\omega_1 \Delta t + b_1), \cdots, \cos(\omega_{d_t} \Delta t + b_{d_t})\right], \tag{2}$$

where $\cos(\cdot)$ is the cosine function. And $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_{d_t}]$ and $\boldsymbol{b} = [b_1, \cdots, b_{d_t}]$ are learnable weights and bias of linear transformation for the time interval $\Delta t$.

## 4.2 Memory Mechanism of Living Habits

Most of the previous self-attentive recommenders [4, 7, 13] focus more on instant intentions while neglecting users' historical living habits. However, maintaining the users' living habits can not only capture the periodicity but also alleviate the negative impact of noise signals as illustrated in Figure 1(a). Inspired by the ability of memory networks [1, 25] in NLP and CTR, we propose the *memory mechanism*. Building upon the accurate instant intentions through TP and TE, the memory mechanism captures the users' living habits by memorizing a series of instant intentions. In general, it consists of two parts: (1) *batch memory updating*, which updates the memory states when new interactions occur; (2) *smooth memory reading*, which reads out the memory states in a smooth manner at a specific time.

*4.2.1* ***Batch Memory Updating.*** The memory mechanism represents each user's (item's) history in a compressed format, thus having the capability to memorize the real intentions from living habits. We denote a memory state as $\boldsymbol{s}_u^t$ $(\boldsymbol{s}_p^t) \in \mathbb{R}^d$ for each user $u$ (item $i$) at time $t$. Each memory state is initialized by an all-zero vector, updated when new interaction occurs, and read out for input construction. All the memory states for users and items can be formulated as $\mathrm{S} = [\mathrm{S}_{\mathcal{U}}; \mathrm{S}_{\mathcal{I}}] \in \mathbb{R}^{|\mathcal{U} \cup \mathcal{I}| \times d}$.

In the following, we will introduce the memory updating and reading for user $u$ at time $t$ as an example, and the same procedure applies to item $i$. When new interaction occurs, the corresponding memory state of the user/item undergoes batch processing for updating. Motivated by the message passing mechanism [9], the updating operation can be divided into three steps: *Message construction*, *Recent message aggregation*, and *State evolution*.

(i) *Message construction.* To store the interaction contexts, given a new interaction $(u, i, t)$, a message is constructed to update state $\boldsymbol{s}_u^t$ of user $u$ in the memory at time $t$, which can be expressed as:

$$\mathbf{m}_u^t = \left[\boldsymbol{s}_u^{t^-} \oplus \boldsymbol{s}_i^{t^-} \oplus \phi(\Delta t)\right], \tag{3}$$

where $\oplus$ is the vector concatenate operator. $\boldsymbol{s}_u^{t^-}$ is the latest state of user $u$ before time $t$, $\Delta t$ denotes the last updated time interval between states $\boldsymbol{s}_u^{t^-}$ and $\boldsymbol{s}_i^{t^-}$, $\phi(\cdot)$ represents the temporal encoding as mentioned in Section 4.1.2.

(ii) *Recent message aggregation.* As each user $u$ may have multiple interactions within a short time interval, generating multiple messages, we can obtain a list of messages $[\mathbf{m}_u^{t_a}, \cdots, \mathbf{m}_u^t]$ in the short time interval $[t_a, t]$. To accurately capture the instant intention $\overline{\mathbf{m}}_u^t$ from these messages, we aim to aggregate them as:

$$\overline{\mathbf{m}}_u^t = \psi_{\text{agg}}\left([\mathbf{m}_u^{t_a}, \cdots, \mathbf{m}_u^t], \epsilon\right), \tag{4}$$

where $\epsilon$ is a hyper-parameter to control the temporal window size. $\psi_{\text{agg}}(\cdot)$ is the *temporal pooling* function as discussed in Section 4.1.1 to alleviate the noisy signal. Therefore, we can obtain a more accurate instant intention from messages.

(iii) *State evolution.* To maintain the temporal evolution of a series of instant intentions and memorize the users' living habits, the state $\boldsymbol{s}_u^t$ of user $u$ at time $t$ is updated upon $\overline{\mathbf{m}}_u^t$ and its previous state $\boldsymbol{s}_u^{t^-}$, which can be formulated as:

$$\boldsymbol{s}_u^t = \text{Evo}\left(\boldsymbol{s}_u^{t^-}, \overline{\mathbf{m}}_u^t\right), \tag{5}$$

where Evo$(\cdot)$ represents a step of the time series function, and we use the GRU [3] cell in this paper. The memory updating for item $i$ has the same process.

*4.2.2* ***Smooth Memory Reading***. The original readout process for user $u$ at time $t$ can be written as:

$$\boldsymbol{h}_u^{(0),t} = \text{Read}(u, t) = \boldsymbol{s}_u^t. \tag{6}$$

However, if the new interaction $(u, i, t)$ in message construction is the noisy signal, the user's readout embedding $\boldsymbol{h}_u^{(0),t}$ is inaccurate at time $t$. To alleviate this issue, we aim to smoothly readout for user $u$ at time $t$ from a list of $u$'s historical memory states $[\boldsymbol{s}_u^{t_b}, \cdots, \boldsymbol{s}_u^t]$ at the time interval $[t_b, t]$. So we rewrite the Eq. (6) as:

$$\boldsymbol{h}_u^{(0),t} = \text{Read}(u, t) = \psi_{\text{read}}\left([\boldsymbol{s}_u^{t_b}, \cdots, \boldsymbol{s}_u^t], \eta\right), \tag{7}$$

where $\eta$ is a hyper-parameter to control the temporal window size. $\psi_{\text{read}}(\cdot)$ is the *temporal pooling* function as mentioned in Section 4.1.1. We denote $\boldsymbol{h}_u^{(0),t}$ as the input information of the first *temporal graph transformer* (TGT) layer, which will be introduced later in Section 4.3. In general, we define the hidden embedding $\boldsymbol{h}_u^{(\ell-1),t} \in \mathbb{R}^d$ as the input to the $\ell$-th layer. For $\ell > 1$, the hidden embedding is generated from the previous layer; when $\ell = 1$, it is read out from the memory states using Eq. (7).

## 4.3 Temporal Graph Transformer Layer

To capture the intentional periodicity, we propose the temporal graph transformer (TGT) layer, which aggregates the temporal information from multi-hop neighbors. Here, we illustrate the calculation process for user $u$ at time $t$ as an example.

*4.3.1* ***Construction of Query, Key and Value***. The query vector at $\ell$-th TGT layer is constructed by the user hidden embedding $\boldsymbol{h}_u^{(\ell-1),t} \in \mathbb{R}^d$ and temporal encoding $\phi(t-t) \in \mathbb{R}^{d_t}$. Then, we have the query vector $\boldsymbol{q}^{(\ell),t} \in \mathbb{R}^{d+d_t}$:

$$\boldsymbol{q}^{(\ell),t} = \left[\boldsymbol{h}_u^{(\ell-1),t} \oplus \phi(t-t)\right], \tag{8}$$

where $\oplus$ is the concatenate operation and the layer number $\ell = \{1, \cdots, L\}$. In addition to the query vector, we also construct the key and value from user $u$'s temporal neighbors $(j_1, t_1), (j_2, t_2), \cdots, \in \mathcal{N}_u^t$. Similar to the construction of the query vector, the key/value matrix for all user $u$'s temporal neighbors can be formulated as:

$$\mathbf{K}^{(\ell),t} = \mathbf{V}^{(\ell),t} = \begin{bmatrix} \boldsymbol{h}_{j_1}^{(\ell-1),t_1} \oplus \phi(t-t_1), \\ \boldsymbol{h}_{j_2}^{(\ell-1),t_2} \oplus \phi(t-t_2), \\ \cdots \end{bmatrix}, \tag{9}$$

where $\mathbf{K}^{(\ell),t}, \mathbf{V}^{(\ell),t} \in \mathbb{R}^{|\mathcal{N}_u^t| \times (d+d_t)}$.

*4.3.2* ***Temporal Self-Attention***. Then, we adopt a self-attention mechanism to obtain the hidden user (item) embeddings and utilize the residual connection to prevent over-smoothing [11] as follows:

$$\tilde{\boldsymbol{h}}_u^{(\ell),t} = \left(\mathbf{W}_v^{(\ell)}\mathbf{V}^{(\ell),t}\right) \cdot \sigma\left(\frac{[\mathbf{W}_k^{(\ell)}\mathbf{K}^{(\ell),t}]^\top [\mathbf{W}_q^{(\ell)}\boldsymbol{q}^{(\ell),t}]}{\sqrt{d+d_t}}\right), \tag{10}$$

$$\boldsymbol{h}_u^{(\ell),t} = \text{MLP}(\boldsymbol{h}_u^{(\ell-1),t} \oplus \tilde{\boldsymbol{h}}_u^{(\ell),t}), \tag{11}$$

where $\sigma(\cdot)$ is the activation function, and we use the softmax function in this part. $\mathbf{W}_q^{(\ell)}, \mathbf{W}_k^{(\ell)}, \mathbf{W}_v^{(\ell)} \in \mathbb{R}^{d \times (d+d_t)}$ are learnable transformation matrices at $\ell$-th layer. By stacking $L$ layers, we can obtain the final embedding for user $u$ as $\boldsymbol{h}_u^t = \boldsymbol{h}_u^{(L),t}$. Analogously, for item $i$, we need to alternate the user information to item information and change the neighbor information in Eq. (8), (9) and (10) according to user-item pairs. Thus, $\boldsymbol{h}_i^t$ for item $i$ can also be calculated.

## 4.4 Multi-task Training

*4.4.1* ***Prediction and BPR Loss***. To optimize parameters, we use the pairwise BPR loss [27] for the top-K recommendation. The objective function of the BPR loss can be formulated as:

$$\mathcal{L}_{BPR} = \sum_{u \in \mathcal{U}} \sum_{(u,i,i',t) \in O} -\ln \sigma\left(\langle \boldsymbol{h}_u^t, \boldsymbol{h}_i^t \rangle - \langle \boldsymbol{h}_u^t, \boldsymbol{h}_{i'}^t \rangle\right), \tag{12}$$

where $y_{u,i}^t = \langle \boldsymbol{h}_u^t, \boldsymbol{h}_i^t \rangle$ denotes the prediction score between user $u$ and item $i$ at time $t$ in triple $(u, i, t)$, and $\langle \cdot, \cdot \rangle$ is the similarity function such as inner product and MLP operation. $O = \{(u, i, i', t) \mid e_{u,i}^t \in \mathcal{E}, e_{u,i'}^t \notin \mathcal{E}\}$ denotes the pairwise training data, and the activation function $\sigma(\cdot)$ is the sigmoid function in this part.

*4.4.2* ***Denoising Task***. The BPR training strategy in Eq. (12) involves positive and negative samples represented by $\boldsymbol{h}_i^t$ and $\boldsymbol{h}_{i'}^t$, respectively. The objective is to make the final user embedding $\boldsymbol{h}_u^t$ close to $\boldsymbol{h}_i^t$ (positive sample) while ensuring it is far away from $\boldsymbol{h}_{i'}^t$ (negative sample). However, if $(u, i, t)$ is a noisy signal, the final user embedding $\boldsymbol{h}_u^t$ will be closer to the noise embedding $\boldsymbol{h}_i^t$ under this training strategy. Therefore, we aim to construct new positive and negative samples. Specifically, the positive sample $\boldsymbol{p}_{u,pos}^t$ for user $u$ at time $t$ can be constructed by a list of memory states from its temporal neighbors $[j, \cdots, i] \in \mathcal{N}_u^t$ as:

$$\boldsymbol{p}_{u,pos}^t = \psi_{\text{pos}}\left([\boldsymbol{s}_j^{t^-}, \cdots, \boldsymbol{s}_i^{t^-}], \lambda\right), \tag{13}$$

where $\lambda$ is a hyper-parameter to control the temporal window size. $\psi_{\text{pos}}(\cdot)$ is the *temporal pooling* function to alleviate the noisy signals. To create a challenging scenario, the negative sample should be related to user $u$ but distinct from the positive sample. Hence, the

**Table 1: Statistics of the Datasets.**

| Dataset | #User | #Item | #Instance | Density |
|---|---|---|---|---|
| Amazon-L&B | 3,819 | 1,581 | 34,278 | 0.23% |
| Gowalla-Food | 15,058 | 26,594 | 553,121 | 0.06% |
| Meituan | 17,862 | 26,236 | 649,101 | 0.07% |

negative sample $\boldsymbol{p}_{u,neg}^t$ is constructed using all user $u$'s historical neighbors within time interval $[0, t)$ as:

$$\boldsymbol{p}_{u,neg}^t = \text{Mean}\left(\boldsymbol{s}_j^{t^-} \mid j \in \mathcal{N}_u^t\right). \tag{14}$$

To make the final output embedding $\boldsymbol{h}_u^t$ more similar to positive sample $\boldsymbol{p}_{u,pos}^t$ against negative sample $\boldsymbol{p}_{u,neg}^t$, we rewrite the Eq. (12) as:

$$\mathcal{L}_{D,\mathcal{U}} = \sum_{u\in\mathcal{U}}\sum_{t} -\sigma\left(\langle \boldsymbol{h}_u^t, \boldsymbol{p}_{u,pos}^t\rangle - \langle \boldsymbol{h}_u^t, \boldsymbol{p}_{u,neg}^t\rangle\right), \tag{15}$$

where the activation function $\sigma(\cdot)$ is the softplus function in this part. In a similar way, the loss for all the items can be obtained as $\mathcal{L}_{D,\mathcal{I}}$. Finally, the complete objective function of the denoising task $\mathcal{L}_D$ is the summation of the above two losses:

$$\mathcal{L}_D = \mathcal{L}_{D,\mathcal{U}} + \mathcal{L}_{D,\mathcal{I}}. \tag{16}$$

In the end, we unify the BPR loss $\mathcal{L}_{BPR}$, denoising task $\mathcal{L}_D$, and the memory regularization for multi-task training. The hybrid objective function is defined as follows:

$$\mathcal{L} = \mathcal{L}_{BPR} + \alpha\mathcal{L}_D + \beta\mathcal{L}_M, \tag{17}$$

where $\mathcal{L}_M = \|\mathrm{S}\|_2$ denotes the $\mathrm{L}_2$ normalization for memory regularization to prevent over-fitting. $\alpha$ and $\beta$ are hyper-parameters to control the relative effect of the denoising task and memory regularization respectively.

## 5 EXPERIMENT

In this section, we conduct experiments on three real-world datasets for life service recommendation to evaluate our proposed method, with the purpose of answering the following three research questions,

- **RQ1:** How does the proposed DPGN perform compared with state-of-the-art methods for life service recommendation?
- **RQ2:** Can DPGN alleviate the noisy signals and capture the intentional periodicity precisely?
- **RQ3:** What is the effect of different components in DPGN?

## 5.1 Experimental Setups

*5.1.1 Datasets.* We evaluate the recommendation performance on two public datasets and a real-world industrial dataset for life service recommendation. Table 1 summarizes the basic statistics of the three datasets. Furthermore, for each dataset, we chronologically split for training/validation/testing in an 8:1:1 ratio. The details of the three datasets are introduced as follows.

- **Amazon-L&B**[4]. This dataset [21] is widely used for recommendation research, which is collected from Amazon. We use the review data of the Luxury and Beauty service from September 20, 2005 to September 24, 2018, and filter out users and items with less than 5 interactions.
- **Gowalla-Food**[5]. This dataset [17] is a location-based service network for recommendation research, which is collected from Gowalla. We use the check-in data of the food service from January 21, 2009 to March 11, 2010, and filter out users and items with less than 10 interactions.
- **Meituan**. This is an industrial dataset collected from one of the largest life service platforms (Meituan) in China. We sample the data of the food delivery service from February 14 to March 28, 2022. User behaviors (100% click and 12% purchase) are recorded in the dataset. Each item belongs to a category. Click data is used to conduct experiments, purchase data, and item categories are used to analyze, and the 10-core setting is also adopted to filter out inactive users and items.

*5.1.2 Baselines.* We compare our DPGN with four categories of baseline methods in the following.

**Graph collaborative filtering methods**:

- **NGCF** [38] adopts GCN layers on the user-item interaction graph to refine user and item representations.
- **LightGCN** [6] simplifies the design of GCN by discarding the nonlinear feature transformations for recommendation.
- **SGL** [40] designs different graph views to denoise noisy signals in implicit feedback. We choose the edge dropout version as SGL-ED.

**Sequential methods**:

- **Caser** [31] combines CNNs and a latent factor model to learn users' sequential and general representations.
- **SASRec** [7] adopts a self-attention mechanism to identify important items from a user's behavior history.
- **TiSASRec** [13] encodes continuous time interval information between interacted items to facilitate the self-attention computation.

**Dynamic graph embedding methods**:

- **JODIE** [10] employs two recurrent neural networks and a novel projection operator to estimate the embedding of a node at any time in the future. We replace its raw BCE loss with the BPR loss for recommendation.
- **DyRep** [34] adopts the temporal point process and posits representation learning as a latent mediation process. The training task of DyRep is the same as JODIE.

**Periodic-related methods**:

- **NARM** [12] adopts a global encoder and a local encoder to capture the user's long-term and short-term preferences.
- **TGSRec** [4] encodes the time intervals for periodicity and combines temporal collaborative filtering with sequential patterns.
- **SLRC** [36] introduces the Hawkes process into collaborative filtering to capture short-term and life-time effects for repeat consumption. We choose the BPR version as SLRC-BPR.
- **TGN** [28] proposes a unified framework of temporal graph networks to capture the temporal evolution of nodes. We choose the attention version as TGN-attn. The training task of TGN-attn is the same as JODIE.

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

[5]http://www.yongliu.org/datasets.html

**Table 2: Comparison results on three datasets. Underline means the best baseline, and bold means the best performance. We report the performance of average and standard deviation over 3 independent runs.**

| Model | Amazon-L&B | | | Gowalla-Food | | | Meituan | | |
|---|---|---|---|---|---|---|---|---|---|
| | Hit@10 | NDCG@10 | MRR | Hit@10 | NDCG@10 | MRR | Hit@10 | NDCG@10 | MRR |
| NGCF | 0.0493$_{\pm0.0034}$ | 0.0209$_{\pm0.0011}$ | 0.0202$_{\pm0.0011}$ | 0.0323$_{\pm0.0004}$ | 0.0105$_{\pm0.0001}$ | 0.0107$_{\pm0.0001}$ | 0.0361$_{\pm0.0006}$ | 0.0176$_{\pm0.0001}$ | 0.0214$_{\pm0.0001}$ |
| LightGCN | 0.0548$_{\pm0.0025}$ | 0.0222$_{\pm0.0010}$ | 0.0208$_{\pm0.0013}$ | 0.0337$_{\pm0.0001}$ | 0.0109$_{\pm0.0001}$ | 0.0111$_{\pm0.0001}$ | 0.0380$_{\pm0.0002}$ | 0.0179$_{\pm0.0001}$ | 0.0216$_{\pm0.0001}$ |
| SGL-ED | 0.0670$_{\pm0.0029}$ | 0.0263$_{\pm0.0013}$ | 0.0243$_{\pm0.0011}$ | 0.0462$_{\pm0.0002}$ | 0.1100$_{\pm0.0001}$ | 0.0112$_{\pm0.0001}$ | 0.0479$_{\pm0.0002}$ | 0.0265$_{\pm0.0001}$ | 0.0266$_{\pm0.0001}$ |
| Caser | 0.0437$_{\pm0.0059}$ | 0.0200$_{\pm0.0026}$ | 0.0181$_{\pm0.0019}$ | 0.0444$_{\pm0.0023}$ | 0.0230$_{\pm0.0011}$ | 0.0231$_{\pm0.0011}$ | 0.0730$_{\pm0.0032}$ | 0.0380$_{\pm0.0022}$ | 0.0376$_{\pm0.0019}$ |
| SASRec | 0.0440$_{\pm0.0109}$ | 0.0218$_{\pm0.0051}$ | 0.0246$_{\pm0.0038}$ | 0.0417$_{\pm0.0009}$ | 0.0229$_{\pm0.0002}$ | 0.0232$_{\pm0.0005}$ | 0.0806$_{\pm0.0015}$ | 0.0417$_{\pm0.0011}$ | 0.0398$_{\pm0.0009}$ |
| TiSASRec | 0.0442$_{\pm0.0181}$ | 0.0230$_{\pm0.0100}$ | 0.0255$_{\pm0.0079}$ | 0.0396$_{\pm0.0017}$ | 0.0214$_{\pm0.0011}$ | 0.0216$_{\pm0.0009}$ | 0.0793$_{\pm0.0007}$ | 0.0409$_{\pm0.0001}$ | 0.0390$_{\pm0.0002}$ |
| JODIE | 0.0556$_{\pm0.0082}$ | 0.0291$_{\pm0.0039}$ | 0.0282$_{\pm0.0031}$ | 0.0843$_{\pm0.0046}$ | 0.0489$_{\pm0.0062}$ | 0.0448$_{\pm0.0042}$ | 0.0471$_{\pm0.0030}$ | 0.0233$_{\pm0.0018}$ | 0.0237$_{\pm0.0015}$ |
| DyRep | 0.0413$_{\pm0.0124}$ | 0.0246$_{\pm0.0066}$ | 0.0256$_{\pm0.0053}$ | 0.0901$_{\pm0.0085}$ | 0.0536$_{\pm0.0056}$ | 0.0488$_{\pm0.0051}$ | 0.0435$_{\pm0.0026}$ | 0.0213$_{\pm0.0019}$ | 0.0215$_{\pm0.0015}$ |
| NARM | 0.0638$_{\pm0.0066}$ | 0.0344$_{\pm0.0020}$ | 0.0309$_{\pm0.0023}$ | 0.0794$_{\pm0.0052}$ | 0.0442$_{\pm0.0031}$ | 0.0419$_{\pm0.0026}$ | 0.0804$_{\pm0.0041}$ | 0.0408$_{\pm0.0031}$ | 0.0395$_{\pm0.0029}$ |
| TGSRec | 0.0559$_{\pm0.0042}$ | 0.0283$_{\pm0.0014}$ | 0.0253$_{\pm0.0005}$ | 0.1595$_{\pm0.0163}$ | 0.1071$_{\pm0.0205}$ | 0.1006$_{\pm0.0213}$ | 0.0619$_{\pm0.0007}$ | 0.0315$_{\pm0.0006}$ | 0.0296$_{\pm0.0006}$ |
| SLRC-BPR | 0.0454$_{\pm0.0010}$ | 0.0276$_{\pm0.0016}$ | 0.0249$_{\pm0.0015}$ | 0.1837$_{\pm0.0014}$ | 0.1390$_{\pm0.0012}$ | 0.0974$_{\pm0.0009}$ | $\underline{0.1053}_{\pm0.0010}$ | $\underline{0.0516}_{\pm0.0011}$ | $\underline{0.0452}_{\pm0.0004}$ |
| TGN-attn | $\underline{0.1196}_{\pm0.0174}$ | $\underline{0.0605}_{\pm0.0070}$ | $\underline{0.0519}_{\pm0.0036}$ | $\underline{0.2440}_{\pm0.0046}$ | $\underline{0.1692}_{\pm0.0064}$ | $\underline{0.1555}_{\pm0.0068}$ | 0.0902$_{\pm0.0037}$ | 0.0468$_{\pm0.0024}$ | 0.0437$_{\pm0.0025}$ |
| **DPGN** | **0.1541**$_{\pm0.0105}$ | **0.0800**$_{\pm0.0072}$ | **0.0695**$_{\pm0.0070}$ | **0.3030**$_{\pm0.0035}$ | **0.2292**$_{\pm0.0028}$ | **0.2144**$_{\pm0.0027}$ | **0.1115**$_{\pm0.0051}$ | **0.0586**$_{\pm0.0035}$ | **0.0542**$_{\pm0.0032}$ |
| Improv. | (+0.0345) | (+0.0150) | (+0.0176) | (+0.0590) | (+0.0600) | (+0.0589) | (+0.0195) | (+0.0118) | (+0.0105) |

*5.1.3 Evaluation Protocols.* We choose several widely used metrics for top-K recommendation: (1) Hit@K (Hit Ratio@K) is a recall metric; (2) NDCG@K (Normalized Discounted Cumulative Gain@K) is a ranking metric; (3) MRR (Mean Reciprocal Rank) is also a ranking metric. And we set $K = 10$ in the experiments. To ensure the reliability of the results, we perform a full ranking with all item candidates and report the average results across all the interactions in the test set [6].

*5.1.4 Model Implementations.* We implement our DPGN framework with PyTorch [23] and perform experiments on Tesla V100 (32GB). We use Adam optimizer to learn parameters, where the batch size is fixed at 200. The node dimension $d$ and time dimension $d_t$ are searched from [64, 128, 256]. We tune the learning rate in [1e-3, 1e-4, 1e-5], search the hyper-parameter $\epsilon$ from [1, 2, 3, 4], search $\eta$ from [1, 2, 3, 4], and search $\lambda$ from [1, 2, 3, 4, 5]. We also search the coefficient $\alpha$ in [0.1, 0.2, 0.3, 0.4, 0.5], and search the regularization coefficient $\beta$ from [5e-4, 1e-4, 5e-5, 1e-5].

## 5.2 Overall Performance (RQ1)

Table 2 illustrates the results of the three datasets. From the results, we have the following observations:

- **Our proposed method consistently achieves the best performance.** In particular, DPGN improves over the strongest baseline w.r.t Hit@10 by 0.0345, 0.0590, and 0.0195 on three datasets, respectively. DPGN also significantly outperforms other methods w.r.t NDCG@10 and MRR. The superiority of DPGN is a result of several factors: (1) DPGN alleviates the noisy signals and models the instant intentions accurately via temporal pooling (TP) and temporal encoding (TE); (2) DPGN captures the users' living habits via memory mechanism; (3) DPGN stacks temporal graph transformer (TGT) layers to aggregate temporal information; (4) DPGN utilizes the denoising task to further alleviate the noisy signals.
- **Periodic-related methods generally outperform other types of methods.** For example, TGN-attn is the best baseline on Amazon-L&B and Gowalla-Food datasets, while SLRC-BPR is the best baseline on Meituan dataset. And both TGN-attn and SLRC-BPR are periodic-related methods. The reason is that these methods focus more on modeling users' historical living habits, which contains a wealth of user intentions. This indicates that capturing the users' living habits is crucial for life service recommendation. However, the user's historical behavior always contains noisy signals, which will have a negative impact on these methods.
- **It is essential to alleviate the noisy signals for life service recommendation.** Compared with NGCF and LightGCN, SGL-ED constructs multi-view contrastive learning via static graph augmentation operations to alleviate the noisy signals. This leads to an improvement of SGL-ED over LigthGCN w.r.t Hit@10 by 0.0125 on the Gowalla-Food dataset. Additionally, SGL-ED even performs better than sequential methods and dynamic graph embedding methods on Amazon L&B dataset, which validates that the time-sensitive methods are more severely affected by noisy signals. Therefore, in DPGN, we employ TP to alleviate the noisy signal for instant intention modeling and subsequently capture the users' living habits by memorizing a series of instant intentions.

## 5.3 Denoising Periodic Modeling (RQ2)

*5.3.1 Statistics of the intention prediction.* As shown in Figure 1(b) in Introduction, we also observe that categories with high click numbers have high purchase conversion rates in sub-sequences. This indicates that the category-level click numbers can serve as an indicator of user purchase intentions. Therefore, in order to assess the prediction accuracy of purchase intentions, we propose a new metric named **IH@K** (Intention-Hit-Ratio@K), which extends the

Table 3: Results of ablation study with model variants for DPGN on three datasets.

| Method | Amazon-L&B | | | Gowalla-Food | | | Meituan | | |
|---|---|---|---|---|---|---|---|---|---|
| | Hit@10 | NDCG@10 | MRR | Hit@10 | NDCG@10 | MRR | Hit@10 | NDCG@10 | MRR |
| (0) **Default** | **0.1684** | **0.0900** | **0.0790** | **0.3078** | **0.2332** | **0.2182** | **0.1186** | **0.0635** | **0.0586** |
| (1) $\psi_{\text{agg}}$ (recent→all) | 0.1377 | 0.0697 | 0.0620 | 0.3003 | 0.2241 | 0.2092 | 0.0665 | 0.0353 | 0.0333 |
| (2) $\psi_{\text{read}}$ (smooth→origin) | 0.1263 | 0.0669 | 0.0608 | 0.2938 | 0.2148 | 0.1991 | 0.0665 | 0.0334 | 0.0321 |
| (3) $\phi$ (temporal→position) | 0.1491 | 0.0756 | 0.0651 | 0.2553 | 0.1828 | 0.1699 | 0.1151 | 0.0604 | 0.0558 |
| (4) w/o memory | 0.0792 | 0.0502 | 0.0460 | 0.2135 | 0.1513 | 0.1415 | 0.0633 | 0.0327 | 0.0313 |
| (5) w/o $\mathcal{L}_{D,\mathcal{I}}$ (item-side) | 0.1548 | 0.0794 | 0.0701 | 0.2829 | 0.2120 | 0.1988 | 0.1160 | 0.0606 | 0.0555 |
| (6) w/o $\mathcal{L}_{D,\mathcal{U}}$ (user-side) | 0.1327 | 0.0724 | 0.0647 | 0.3047 | 0.2307 | 0.2160 | 0.0881 | 0.0454 | 0.0428 |

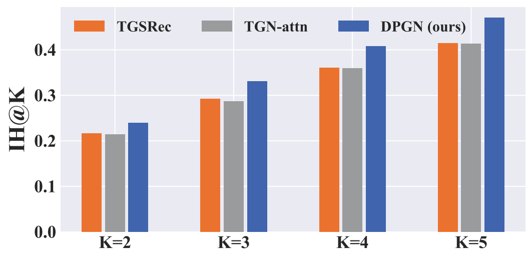


**Figure 3: IH@K (Intention-Hit-Ratio@K) for intention prediction on Meituan dataset. The y-axis represents the values of IH@K while the x-axis corresponds to the prediction results of different K and methods.**

commonly used metric Hit@K (Hit-Ratio) as:

$$IH@K = \frac{C_{\text{hit}}@K}{|\mathcal{E}_P|} \tag{18}$$

where $|\mathcal{E}_P|$ denotes the number of purchase interactions, $C_{\text{hit}}@K$ denotes the count of hit categories in top-K category-level ranking lists. Note that, to generate the category-level ranking list, we group the categories together and sum their prediction scores.

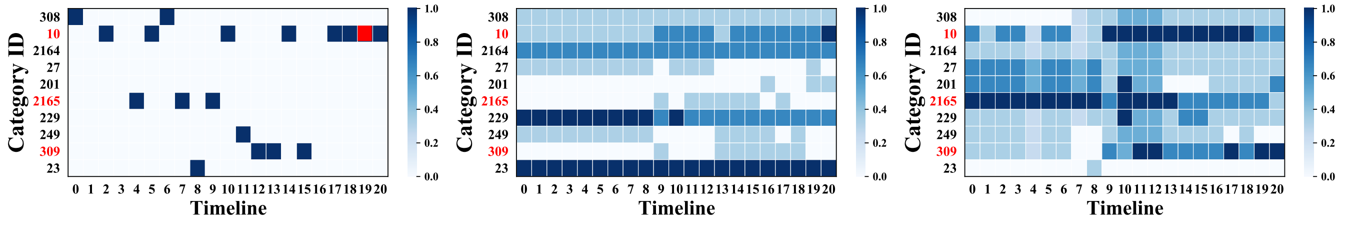
Then, we made statistics of IH@K for different methods on Meituan dataset. As shown in Figure 3, we can observe that our DPGN consistently achieves higher IH@K values compared to TGSRec and TGN-attn, indicating that recommendations made by DPGN align more closely with the user's real purchase intentions. The success of DPGN in denoising during intentional periodicity modeling contributes to this improvement. Furthermore, TGN-attn outperforms TGSRec w.r.t Hit@K on Meituan dataset as shown in Table 2, but the IH@K values of TGSRec and TGN-attn are nearly identical. Besides, TGSRec even outperforms TGN-attn w.r.t IH@K when $K \leq 3$. This indicates that both TGSRec and TGN are affected by noisy signals during intention modeling. All the following observations emphasize the importance of alleviating the noisy signals and capturing the intentional periodicity for life service recommendation.

*5.3.2 Case study.* To further investigate the intentional periodicity modeling with noisy signals, we conduct a case study to visualize the category-level prediction accuracy in a randomly selected sub-sequence as shown in Figure 4. From the ground truth in Figure 4(a), we can observe that the category IDs with top-3 click numbers are 10 (Fast food), 2165 (Beverage shop), and 309 (Western food), in which the user's final purchase category ID is 10 at timeline 19. This verifies that categories with high click numbers have higher purchase conversion rates. Specifically, from the results in Figure 4(b) and (c), we can observe that compared to TGN-attn, our DPGN accurately predicts the correct categories at the appropriate times within the sub-sequence, which validates that DPGN models the user's intentions more accurately at different times.

## 5.4 Model Analysis (RQ3)

*5.4.1 Ablation study with model variants.* We further conduct an ablation study with several variants to validate the contributions of key components in DPGN: functions in instant intention modeling (1-3), memory mechanism (4), and denoising task (5-6). As for instant intention modeling, (1) replace TP with all average aggregation for $\psi_{\text{agg}}(\cdot)$; (2) replace TP with original read operation for $\psi_{\text{read}}(\cdot)$. (3) replace TE with position encoding for $\phi(\cdot)$ in *Message construction* and TGT layer. As for the memory mechanism, (4) not using the whole memory mechanism and only use single ID embedding as input. As for the denoising task, (5) not using the denoising task for all items; (6) not using the denoising task for all users. Table 3 reports the performance of these variants on three datasets. Here, we can make the following observations.

- **Effectiveness of functions in instant intention modeling**. **First**, let's analyze variants (1) and (2) together. As for temporal pooling (TP), both $\psi_{\text{agg}}(\cdot)$ and $\psi_{\text{read}}(\cdot)$ aim to extract the most representative information shared by recent behaviors to alleviate the noisy signals for instant intentions. The former is used to smooth the messages, while the latter is used to smooth the readout memory states. The results show that variants (1) and (2) exhibit a decrease in performance across all three datasets, which confirms the importance of temporal pooling in alleviating noisy signals. **Second**, variant (3) suffers severe performance degradation on Amazon-L&B and Gowalla-Food dataset. This suggests that temporal encoding (TE) plays a crucial role in capturing temporal dynamics and periodicity, which is essential for accurate instant intention modeling. All the following observations indicate that TP and TE can alleviate the noisy signals and model the instant intention more accurately for life service recommendation.



(a) Ground truth. (b) Prediction of TGN-attn. (c) Prediction of DPGN (ours).

**Figure 4: Case study for the category-level prediction accuracy over time on Meituan dataset. The y-axis shows different category IDs and the x-axis shows the timeline in a sub-sequence. The blue rectangle denotes the predicted score of click behavior and the red rectangle denotes the purchase behavior. The deeper the blue color, the higher the predicted score. (a) represents the ground truth. (b) and (c) represent the predicted scores for a random user.**

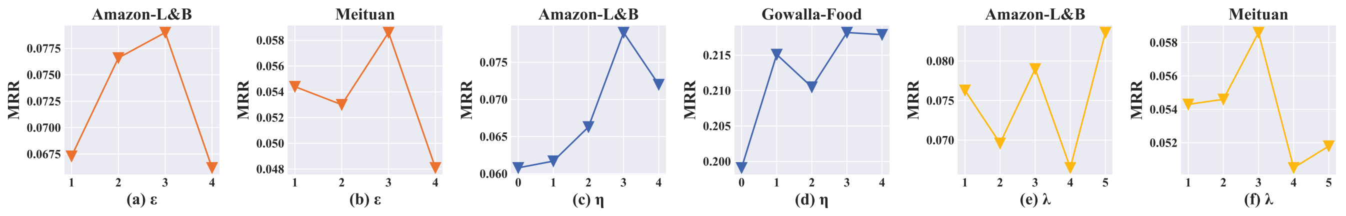


**Figure 5: The MRR of different hyper-parameters (recent window size $\epsilon$, $\eta$, $\lambda$) on three datasets. (a) and (b) represent $\epsilon$, (c) and (d) represent $\eta$, (e) and (f) represent $\lambda$ on different datasets.**

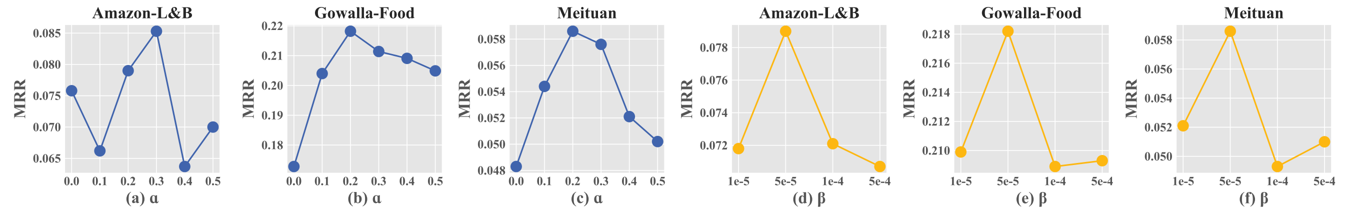


**Figure 6: The MRR of different hyper-parameters (coefficient $\alpha$, $\beta$) on three datasets. (a), (b) and (c) represent $\alpha$, (d), (e) and (f) represent $\beta$ on different datasets.**

- **Effectiveness of memory mechanism of living habits**. The performance of variant (4) exhibits a significant drop of 0.0892, 0.0943, and 0.0553 w.r.t Hit@10 on Amazon-L&B, Gowalla-Food, and Meituan datasets respectively. This clearly demonstrates that capturing the users' living habits with the memory mechanism is essential for life service recommendation.
- **Effectiveness of denoising task**. By examining the variant (5) and (6) together, we can observe that solely relying on either the user-side or item-side denoising task does not yield optimal performance across all three datasets. This demonstrates that the denoising task can alleviate the noisy signals for life service recommendation. Moreover, we notice that $\mathcal{L}_{D,\mathcal{U}}$ is more important on Amazon-L&B and Meituan dataset, while $\mathcal{L}_{D,\mathcal{I}}$ is more important on Gowalla-Food dataset. This indicates that the importance of $\mathcal{L}_{D,\mathcal{U}}$ and $\mathcal{L}_{D,\mathcal{I}}$ varies across different datasets.

*5.4.2 Impact of temporal window size for TP.* To investigate the sensitivity of temporal window size for TP, we mainly focus on three hyper-parameters: $\epsilon$ of $\psi_{\text{agg}}(\cdot)$ in Eq. (4), $\eta$ of $\psi_{\text{read}}(\cdot)$ in Eq. (6), and $\lambda$ of $\psi_{\text{pos}}(\cdot)$ in Eq. (13). Figure 5 reports the performance of these hyper-parameters on three datasets. We can observe that excessively large or small window sizes do not yield optimal performance. This is because a small window size can be influenced by recent noisy signals, while an overly large window size captures instant intentions beyond the current sub-sequence. Hence, the optimal values of three hyper-parameters lie in the middle (i.e. $\epsilon = \eta = \lambda = 3$).

*5.4.3 Impact of the coefficient for multi-task training.* To further investigate the sensitivity of the coefficient, we mainly focus on two hyper-parameters: $\alpha$ of $\mathcal{L}_D$ and $\beta$ of $\mathcal{L}_M$ in Eq. (17). Figure 6 reports the performance of these hyper-parameters on three datasets. When $\alpha = 0$ (no denoising task), the performance drops on all three datasets, emphasizing the importance of alleviating noisy signals. Larger $\alpha$ values result in over-smoothing and do not achieve optimal performance. The optimal value of $\alpha$ is around 0.2 or 0.3. Moreover, to prevent over-fitting, we propose memory regularization and find the optimal value setting of its coefficient $\beta$ is 5e-5.

## 6 CONCLUSION

In this paper, we address the challenges of alleviating the noise signals and capturing intentional periodicity for life service recommendation, which has received limited attention in previous research. To tackle these issues, we propose a novel DPGN method. As for instant intention modeling, we propose the temporal pooling to encode the most representative information and the temporal encoding to encode the time intervals. To capture the users' living habits, we propose the memory mechanism to memorize a series of instant intentions. To further capture the intentional periodicity, we propose the temporal graph transformer layer to aggregate temporal information from neighbors. The denoising task is further proposed to alleviate the noisy signals. Extensive experiments on three real-world datasets demonstrate the effectiveness of our proposed DPGN.

## 7 ACKNOWLEDGMENT

This research was supported by the National Natural Science Foundation of China (Grant No: 62106221, 61972349), Zhejiang Provincial Natural Science Foundation of China under Grant No. LTGG23F030005, Ningbo Natural Science Foundation (Grant No: 2022J183).

## REFERENCES

[1] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[2] Huixuan Chi, Hao Xu, Hao Fu, Mengya Liu, Mengdi Zhang, Yuji Yang, Qinfen Hao, and Wei Wu. 2022. Long Short-Term Preference Modeling for Continuous-Time Sequential Recommendation. *arXiv preprint arXiv:2208.00593* (2022).

[3] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS'Workshop*.

[4] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *CIKM*. 433–442.

[5] Yunjun Gao, Yuntao Du, Yujia Hu, Lu Chen, Xinjun Zhu, Ziquan Fang, and Baihua Zheng. 2022. Self-Guided Learning to Denoise for Robust Recommendation. *SIGIR* (2022).

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[8] Ori Katz, Oren Barkan, Noam Koenigstein, and Nir Zabari. 2022. Learning to Ride a Buy-Cycle: A Hyper-Convolutional Model for Next Basket Repurchase Recommendation. In *RecSys*. 316–326.

[9] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *ICLR* (2016).

[10] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. 1269–1278.

[11] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. 2020. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739* (2020).

[12] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-Based Recommendation. In *CIKM*. 1419–1428.

[13] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[14] Xiang Li, Chao Wang, Bin Tong, Jiwei Tan, Xiaoyi Zeng, and Tao Zhuang. 2020. Deep time-aware item evolution network for click-through rate prediction. In *CIKM*. 785–794.

[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.

[16] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.

[17] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *CIKM*. 739–748.

[18] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[19] Lynn H Loomis. 2013. *Introduction to abstract harmonic analysis.*

[20] Jun Ma, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, and Lei Zhao. 2020. Modeling Periodic Pattern with Self-Attention Network for Sequential Recommendation. In *DASFAA*. Springer, 557–572.

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[22] Larry R Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications* 5 (2001), 64–67.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *NIPS* 32 (2019).

[24] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait detection. In *ECIR*. 810–817.

[25] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *SIGIR*. 565–574.

[26] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten De Rijke. 2019. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI*, Vol. 33. 4806–4813.

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[28] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *ICML'Workshop* (2020).

[29] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I Nikolenko. 2020. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In *WSDM*. 528–536.

[30] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM (CIKM '19)*. 1441–1450.

[31] Jiaxi Tang and Ke Wang. 2018. Personalized Top-n Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.

[32] Changxin Tian, Zihan Lin, Shuqing Bian, Jinpeng Wang, and Wayne Xin Zhao. 2022. Temporal Contrastive Pre-Training for Sequential Recommendation. In *CIKM*. 1925–1934.

[33] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. 2022. Learning to Denoise Unreliable Interactions for Graph Collaborative Filtering. In *SIGIR*. 122–132.

[34] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning Representations over Dynamic Graphs. In *ICLR*.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *ICLR* (2017).

[36] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In *WWW*. 1977–1987.

[37] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *WSDM*. 373–381.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *SIGIR* (2019), 165–174. arXiv:1905.08108

[39] Yu Wang, Xin Xin, Zaiqiao Meng, Joemon M Jose, Fuli Feng, and Xiangnan He. 2022. Learning Robust Recommenders through Cross-Model Agreement. In *WWW*. 2015–2025.

[40] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.

[41] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. *AAAI* 33 (2019), 346–353. arXiv:1811.00855

[42] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *ICDE*. IEEE, 1259–1273.

[43] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive Representation Learning on Temporal Graphs. In *ICLR*.

[44] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time matters: Sequential recommendation with complex temporal information. In *SIGIR*. 1459–1468.

[45] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. *AAAI* 33 (2019), 5941–5948.

[46] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. 1059–1068.

[47] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *KDD*. 1079–1088.