Exploiting Positional Information for Session-based Recommendation

RUIHONG QIU, The University of Queensland, Australia
ZI HUANG, The University of Queensland, Australia
TONG CHEN, The University of Queensland, Australia
HONGZHI YIN*, The University of Queensland, Australia

For present e-commerce platforms, it is important to accurately predict users' preference for a timely next-item recommendation. To achieve this goal, session-based recommender systems are developed, which are based on a sequence of the most recent user-item interactions to avoid the influence raised from outdated historical records. Although a session can usually reflect a user's current preference, a local shift of the user's intention within the session may still exist. Specifically, the interactions that take place in the early positions within a session generally indicate the user's initial intention, while later interactions are more likely to represent the latest intention. Such positional information has been rarely considered in existing methods, which restricts their ability to capture the significance of interactions at different positions. To thoroughly exploit the positional information within a session, a theoretical framework is developed in this paper to provide an in-depth analysis of the positional information. We formally define the properties of *forward-awareness* and *backward-awareness* to evaluate the ability of positional encoding schemes in capturing the initial and the latest intention. According to our analysis, existing positional encoding schemes are generally *forward-aware* only, which can hardly represent the dynamics of the intention in a session. To enhance the positional encoding scheme for the session-based recommendation, a dual positional encoding (DPE) is proposed to account for both *forward-awareness* and *backward-awareness*. Based on DPE, we propose a novel Positional Recommender (PosRec) model with a well-designed Position-aware Gated Graph Neural Network module to fully exploit the positional information for session-based recommendation tasks. Extensive experiments are conducted on two e-commerce benchmark datasets, *Yoochoose* and *Diginetica* and the experimental results show the superiority of the PosRec by comparing it with the state-of-the-art session-based recommender models.

CCS Concepts: • **Information systems** → **Recommender systems**.

Additional Key Words and Phrases: session-based recommendation, positional encoding, graph neural network

---

*Corresponding author.

Fig. 1. Illustration of the relationship between positions and intention dynamics in a session. Although the forward counting positions of v_6 in session $\mathcal{S}_1$ and $\mathcal{S}_2$ are the same, their backward counting positions are different, leading to different relative positions in a session.

1 INTRODUCTION

Nowadays, recommender systems (RS) play an essential role in e-commerce platforms. Traditional RS [32–34] predict a user's preference by equally taking the historical interactions into consideration, e.g., clicks of items, listening to songs or watching movies. Generally, a user's preference shifts as time goes on, where the traditional RS are less capable of predicting it. To enable a model to deal with this shift, session-based recommender systems (SBRS) have recently emerged, which predict the users' current preferences based on a session [13–15, 22, 26, 30, 35, 49, 51]. A session is defined as a short sequence of user-item interactions within a certain period.

Although a session is assumed to imply the current preference, interactions happening at different stages in a session usually represent different intentions. On one hand, interactions in earlier positions may reflect the initial intention of a user. On the other hand, interactions closer to the end of a session usually demonstrate a better alignment with the latest intention. Such a difference is illustrated in Fig. 1 with two sample sessions. An item in a certain position in a session carries the positional information that reflects the initial and the latest intention. They are referred to as *forward* and *backward* positional information respectively in this work. A main purpose of our paper is to develop a positional encoding scheme to capture these two types of information.

How to effectively represent these two types of positional information remains a challenge in the session-based recommendation. For models using RNN as encoders [3, 13, 14], interactions are fed in the model according to their time order. These models implicitly make use of the positional information by considering the interactions sequentially. They suffer from easily forgetting the initial intention because the recurrent structure will potentially focus more on recent data. Attention-based approaches [4, 22, 26] apply the self-attention mechanism to compute the session representation. The attention mechanism utilizes the positional information in two ways: (1) including a positional encoding; and (2) using the last interaction in a session to attend to other interactions in the same session. When using the positional encoding [4], it captures the *forward* positional information because the positional encoding determines the position by counting from the beginning of a sequence. While for the latter case [22, 26], it neglects the positional information of all other interactions except for the last one, which merely represents the most recent intention. GNN-based methods [29–31, 47, 49, 51] firstly generate a session graph based on the relative position between interactions and further apply the self-attention to generate session representations. For example, for the session on the left in Fig. 1, there will be a directed edge connecting v_7 to v_3. While for the session on the right in Fig. 1, there will be edges connecting v_7 to both v_5 and v_6. In terms of the *forward* and *backward* positional information, the model cannot tell which item is on the first or the last position. Therefore, the positional information leveraged in the GNN model is rather limited as the constructed session graphs tend to neglect both the *forward* and the *backward* position information.

Specifically for the attention mechanism in sequence modeling, positional encoding is the most widely-used to capture the positional information, which is introduced to represent the absolute position of words in a sentence for natural language processing [42]. It is expected to extract the positional information for words appearing in a specific position counted from the beginning of a sentence. However, in language modeling, the relative positions between words are more important than the absolute positions in a sentence, which makes the original positional encoding deprecated in recent language models [9, 36, 52]. Recent recommender systems that use the attention mechanism usually involve a learnable version of the absolute positional encoding [4, 6, 17, 38]. Similar to the fixed positional encoding, a learnable one can only capture the *forward* positional information as well.

In this paper, the positional information in SBRS is firstly formally defined in terms of the *forward* and the *backward* positional information. Besides, the abilities of models in capturing this information are further analyzed. *Forward-awareness* and *backward-awareness* are mainly investigated as the properties of existing position encoding schemes to represent the positional information. More importantly, based on the theoretical analysis, we propose a novel dual positional encoding scheme, which can capture the positional information with *forward-awareness* and *backward-awareness* in the session-based recommendation. In attention to the dual positional encoding scheme, a well-designed Position-aware Gated Graph Neural Network module is proposed to further incorporate the positional information in the session representation learning.

In summary, the main contributions of this paper are as follows:

- A **theoretical framework** is developed to analyze the ability of different positional encoding schemes in representing the positional information for SBRS.
- A **Dual Positional Encoding (DPE)** scheme is proposed to represent the positional information for SBRS, which can be extended to a learnable version, denoted as LDPE.
- A **Positional Recommender model (PosRec)** is proposed based on (L)DPE, in which a Position-aware Gated Graph Neural Network (PGGNN) module is designed to further exploit the positional information in SBRS.
- **Extensive experiments** are conducted on two real-world benchmark SBRS datasets, *Yoochoose*[1] and *Diginetica*[2]. The empirical results demonstrate the superiority of the proposed PosRec andd (L)DPE compared with baselines.

This paper is structured as follows: in Section 2, the related work about SBRS and the positional encoding is briefly reviewed. In Section 3, the theoretical framework is elaborated, followed by the explanation of the proposed (L)DPE and PosRec model in Section 4. In Section 5, experiments are conducted to evaluate the effectiveness of our method.

2 RELATED WORK

In this section, we review three main topics of previous research: the session-based recommendation, the positional encoding, and the graph neural networks.

2.1 Session-based Recommendation

Markov chain is applied by many models [35, 55] to learn the dependency of items in sequential data. Using probabilistic decision-tree models, Zimdars et al. [55] proposed to encode the state of the transition pattern of items. Shani et al. [35] made use of a Markov Decision Process (MDP) to compute item transition probabilities.

---

[1]https://2015.recsyschallenge.com/challenge.html
[2]http://cikm2016.cs.iupui.edu/cikm-cup/

Deep learning models become popular since the widely use of recurrent neural networks [1, 13, 14, 22–24, 26, 28]. There are three main branches of methods to perform the representation learning of the session, i.e., recurrent models [13, 14, 22], attention models [12, 26, 39] and graph models [29–31, 47, 49–51]. (1) For recurrent models, e.g., GRU4REC [13, 14] and NARM [22], Gated Recurrent Unit [8] and Long Short-Term Memory [16] are applied respectively and the positional information is implicitly modeled by the recurrent computing procedure. The recurrent structure includes a strong inductive bias that the relationship between items is linear along with the position. (2) For attention models, e.g., NARM (a self-attention layer is applied after the recurrent layer) and STAMP [26] utilizes self-attention [42] over the last item to capture the relationship between the last item and the rest in the session. These attention-based methods only consider the importance of the last position while neglecting other positions. (3) In graph modeling, e.g., SR-GNN [49], GC-SAN [51], FGNN [30] and MGNN-SPred [47], a session is converted into a graph and Graph Neural Networks (GNN) [19, 25, 43] captures the connectivity of items. Afterward, a readout function is applied to compute a session representation with the processed item representations. For SR-GNN and GC-SAN, the readout function is similar to attention-based models by performing a self-attention over the last item. While FGNN uses a Set2Set [44] module and computes a descriptive vector, which is considered as a latent description of items. MGNN-SPred makes use of the mean feature of the whole sequence to represent the user modeling. Consequently, these GNN-based methods only capture the relative position for the connected items, which does not satisfy the *forward-awareness* and *backward-awareness*. The proposed PosRec falls into the category of graph-based model. To enhance the exploitation of the positional information of the graph representation learning, the (L)DPE is included in the embedding of the items and the graph neural network is redesigned to have a position-aware module.

Sequential recommendation is a close research field to SBRS. In recent years, deep learning models are very popular [7, 11, 17, 38, 41, 46, 54]. Caser [41] applies convolutional layers to process the embeddings of items in a sequence. SASRec [17] and BERT4Rec [38] use the Transformer [42] in a single direction style and a bidirection style respectively to model the sequential pattern in the interaction sequence.

2.2 Positional Encoding

Absolute positional encoding is firstly introduced with the attention structure to provide the access of sequential information for the permutation invariant computation [42]. It assigns a fixed vector to each position in a sequence. The vector is computed either in a sinusoidal way or a learned style. For example, the language model BERT [10] and the recommendation model BERT4Rec [38], they both use the learned positional encoding. **Relative positional encoding** is later proposed to encode the relative position of two words, which is more meaningful for the natural language [9, 36, 45, 52]. For example, the language model XLNet [52] and Transformer-XL [9] propose different types of relative position encodings to represent the relative positional information between words in a sentence. **Other positional encodings** include different positional encoding schemes that are suitable for data structures other than one-dimensional sequence. For example, to apply the attention to images, there are 2D positional encoding schemes [2, 5, 21, 27, 48] that provide either the absolute or the relative encoding. For example, the attention augmented network [2] designs a 2D relative positional encoding to encode the positional information in the activation map. For tree structures, Shiv and Quirk [37] proposed a specific scheme to encode the relationship between the root node and children nodes.

2.3 Graph Neural Networks

Recently, to enable neural networks to work on structured data (e.g., graph, point cloud, etc.), Graph Neural Networks (GNN) are widely investigated [19, 25, 43, 53]. Generally, the computation flow of GNN is called message passing, which is based on neighborhood aggregation. For example, GCN [19], GAT [43] and GGNN [25] are majorly different in the aggregation method. However, these GNN models could easily fall into a lack of representative ability since the message passing is performed on a narrow scope of nodes. Thus, PGNN [53] is proposed to include the information from randomly chosen anchor nodes to utilize extra structural information.

3 THEORETICAL FRAMEWORK FOR POSITIONAL ENCODING

In this section, we build up the theoretical framework to analyze the property of different positional encoding schemes and what is needed to represent the positional information for SBRS.

3.1 Positional Encoding

The positional encoding (PE) is introduced by [42] to enable the self-attention module to utilize the positional information of languages. Here, the sinusoidal positional encoding (SDE) $P \in \mathbb{R}^{d \times 1}$ of a token at position pos in the session of length l is defined as:

$$
\begin{aligned}
P^l_{pos,2i} &= \sin(pos/f(i)), \\
P^l_{pos,2i+1} &= \cos(pos/f(i)),
\end{aligned}
\tag{1}
$$

where $i \in \{0, 1, \ldots, d/2 - 1\}$, d is the dimension of the feature vector and $f(i) = 10000^{2i/d}$. In the following, all $pos \in \{0, 1, \ldots, l - 1\}$ if not specified.

3.2 Property of Positional Encoding

DEFINITION 3.1 (FORWARD-AWARENESS). *A positional encoding P is forward-aware in positional information if $\forall p, q \in \mathbb{Z}^+, \exists A \subseteq \{0, 1, \ldots, d-1\}, A \neq \varnothing$, for two positions pos_a and pos_b, if $pos_a = pos_b$, then $P^p_{pos_a,A} = P^q_{pos_b,A}$ and if $pos_a \neq pos_b$, then $P^p_{pos_a,A} \neq P^q_{pos_b,A}$.*

DEFINITION 3.2 (BACKWARD-AWARENESS). *A positional encoding P is backward-aware in positional information if $\forall p, q \in \mathbb{Z}^+, \exists B \subseteq \{0, 1, \ldots, d-1\}, B \neq \varnothing$, for two positions pos_a and pos_b, if $p - pos_a = q - pos_b$, then $P^p_{pos_a,B} = P^q_{pos_b,B}$ and if $p - pos_a \neq q - pos_b$, then $P^p_{pos_a,B} \neq P^q_{pos_b,B}$.*

To investigate the representation ability of a PE in a session, we define two features: *forward-awareness* and *backward-awareness*. If a PE is *forward-aware*, the PE of the first token is the same for all sequences. Furthermore, if a position pos exists in any sequence, the PE for pos is the same across these sequences. For example, if we assign the position itself as the PE, i.e., $P^l_0 = 0, P^l_1 = 1 \ldots$, then it is *forward-aware*. In contrast, if a PE is *backward-aware*, the PE of the last token is the same for all sequences. Furthermore, if an h-th last position pos exists in any sequence, the PE for pos is the same across these sequences. For example, if we assign the reverse position as the PE, i.e., $P^l_{l-1} = 0, P^l_{l-2} = 1 \ldots$, then it is *backward-aware*. A demonstration of *forward-awareness* and *backward-awareness* can be found in Fig. 1.

PROPERTY 3.1. *If a positional encoding P is forward-aware, $\forall 0 \leq pos_a \leq pos_b < min(p, q), \exists f : \mathbb{R}^{d \times 1} \times \mathbb{R}^1 \times \mathbb{R}^1 \mapsto \mathbb{R}^{d \times 1}, \exists A, s.t. P^p_{pos_b,A} = f(P^p_{pos_a,A}, pos_a, pos_b), then P^q_{pos_b,A} = f(P^q_{pos_a,A}, pos_a, pos_b).*

PROOF. Following Definition 3.1, because items are at the same position pos_a, $P^p_{pos_a,A} = P^q_{pos_a,A}$. Similarly for position pos_b, $P^p_{pos_b,A} = P^q_{pos_b,A}$. Then $P^q_{pos_b,A} = f(P^q_{pos_a,A}, pos_a, pos_b)$ holds for the function $f(\cdot, \cdot, \cdot)$. □

If there is a mapping between two PE in a session, the mapping also applies to other sessions that contain same positions. Similarly, the property of *backward-aware* PE is as the following:

PROPERTY 3.2. *If a positional encoding P is backward-aware, $\forall 0 \leq pos_a \leq pos_b < p, 0 \leq pos_c \leq pos_d < q, \exists f : \mathbb{R}^{d\times 1} \times \mathbb{R}^1 \times \mathbb{R}^1 \mapsto \mathbb{R}^{d\times 1}, \exists B, s.t. P^p_{p-pos_a,B} = f(P^p_{p-pos_b,B}, pos_a, pos_b),$ if $p - pos_a = q - pos_c$ and $p - pos_b = q - pos_d$, then $P^q_{q-pos_c,B} = f(P^q_{q-pos_d,B}, pos_a, pos_b)$.*

PROOF. Following Definition 3.2, because item at pos_b for length p and item at pos_d for length q are at the reverse position $p - pos_b$, $P^p_{pos_b,B} = P^q_{pos_d,B}$. Similarly for position pos_a and pos_c, $P^p_{pos_a,B} = P^q_{pos_c,B}$. Then $P^q_{pos_c,B} = f(P^q_{pos_d,B}, pos_a, pos_b)$ holds for the function $f(\cdot, \cdot, \cdot)$. □

These Definitions and Properties together give another important Properties of an absolute PE.

PROPERTY 3.3. *An absolute positional encoding is unique for each position.*

PROOF. If there are duplicate PE for different positions, Definition 3.1 and 3.2 are violated. □

3.3 Positional Information for Session-based Recommendation

DEFINITION 3.3. *A positional encoding that can represent the positional information in SBRS is both forward-aware and backward-aware.*

As discussed in the Introduction, the position in a session carries specific positional information in SBRS. The first item reflects the initial intention of the user while the last item is always considered more relevant to the latest preference of the user. And the items in-between usually represent the preference shift inside the session. For the *forward-aware* requirement, following Definition 3.1, two items at the same position of two different sessions always have the same slice of their PE. Following Property 3.1, the relationship between any position and the first position is the same across different sessions. As for the *backward-aware* requirement, the position in *forward-aware* requirement is changed into the reverse position following Definition 3.2 and Property 3.2.

THEOREM 3.4. *The sinusoidal positional encoding cannot represent the positional information in SBRS because it is forward-aware but not backward-aware.*

PROOF. We first prove that SPE is *forward-aware* and then SPE is not *backward-aware*. (1) According to Eq. (1), SPE directly follows Definition 3.1 for *forward-aware*. (2) Take the last item of two sessions w.r.t. length 1 and 2 as example. For length 1 session, $P^1_{0,2i} = 0$ and $P^1_{0,2i+1} = 1$. If SPE is *backward-aware*, for length 2 session, there should be a slice of $P^2_{1,B}$ is the same as $P^1_{0,2i}$ and $P^1_{0,2i+1}$. For $2i$ dimension of SPE, $P^1_{0,2i} = P^2_{1,2i} = \sin(1/10000^{2i/d})$. It is clear that $1/10000^{2i/d} \in [0.00001, 1]$. Then $P^1_{0,2i} \neq P^2_{1,2i}$. Similarly, $P^1_{0,2i+1} \neq P^2_{1,2i+1}$. Therefore, SPE is not *backward-aware*. □

This Theorem states that the sinusoidal positional encoding is not informative for positional information in session-based recommendation. As proved above, SPE is *forward-aware* because it is exactly calculated based on the position. Using SPE in any SBRS can only indicate how far an item is from the user's initial intention. However, SPE is not *backward-aware* as it simply cannot tell if an item is at the last position of a session. In SBRS, it is crucial to know the preference shift within the session [14, 35]. Because SPE is not *backward-aware*, if a model uses SPE, there is no information about the closeness between an item and the user's latest preference (i.e., the item at the last position).

COROLLARY 3.5. *The relative positional encoding cannot represent the positional information in SBRS because it is neither forward-aware nor backward-aware.*

PROOF. The RPE is explored by recent language models [9, 36, 52]. During the attention score calculation, the absolute positional encoding P, e.g., SPE, is included as:

$$A = (X_i + P_i)W_{qry}W_{key}^\top(X_j + P_j)^\top, \tag{2}$$

where X is the input feature and W is trainable weights.

For RPE in different work, they basically follow a format:

$$A = X_iW_{qry}W_{key}^\top X_j^\top + g(P_{ij}), \tag{3}$$

where P only represents the relative position between i and j. Because P_{ij} does not provide any information about the absolute position of a token, for different center tokens i_1 and i_2, $P_{i_1j} \neq P_{i_2j}$. Because j can be before or after i in position, RPE simultaneously is not *forward-aware* and *backward-aware*. □

Relative positional encoding (RPE) is designed to relax the assumption in language models that a word in an absolute position has the same meaning. RPE focuses more on the meaning of the relative position between two words. As proved above, RPE is neither *forward-aware* nor *backward-aware*, thus failing to meet both requirements of SBRS.

Empirically, the closer an item is to the last item, the more accurate it can reflect the user's latest preference. As discussed in the Introduction, many methods consider the last item as the representation of the latest preference (usually referred to as short-term or local preference). Meanwhile, other items are treated with less importance (usually referred to as long-term or global preference). Following Theorem 3.4, SPE only contains the *forward-awareness*. Intuitively, we can modify the SPE to a reverse sinusoidal positional encoding (RSPE):

$$\begin{aligned} P^l_{l-pos-1,2i} &= \sin((l - pos - 1)/f(i)), \\ P^l_{l-pos-1,2i+1} &= \cos((l - pos - 1)/f(i)). \end{aligned} \tag{4}$$

COROLLARY 3.6. *The reverse sinusoidal positional encoding cannot represent the positional information in SBRS because it is backward-aware but not forward-aware.*

PROOF. Similar to the proof of Theorem 3.4, we firstly prove RSPE is *backward-aware* and then is not *forward-aware*. (1) According to Eq. (4), RSPE directly follows Definition 3.2 for *backward-aware*. (2) Take the first item of two sessions w.r.t. length 1 and 2 as example. For length 1 session, $P^1_{0,2i} = 0$ and $P^1_{0,2i+1} = 1$. If RSPE is *forward-aware*, for length 2 session, there should be a slice of $P^2_{0,A}$ is the same as $P^1_{0,2i}$ and $P^1_{0,2i+1}$. For $2i$ dimension of RSPE, $P^1_{0,2i} = P^2_{0,2i} = \sin(1/10000^{2i/d})$. It is clear that $1/10000^{2i/d} \in [0.00001, 1]$. Then $P^1_{0,2i} \neq P^2_{0,2i}$. Similarly, $P^1_{0,2i+1} \neq P^2_{0,2i+1}$. Therefore, RSPE is not *forward-aware*. □

With RSPE rather than SPE, a model is theoretically able to utilize the positional information that can reflect how an item is different from the latest preference in the session. But obviously, RSPE neglects the positional information representing the relationship between an item and the initial intention.

3.4 Beyond Single Directional Positional Encoding

In the content above, we focus on the positional encoding that only rolls out in a single direction. In the following, we will discuss the additional positional encoding and 2D positional encoding.

Direct addition of SPE and RSPE fails in this situation. Such an addition will create a symmetric positional encoding that for pos_a and pos_b in a length l session, if $pos_a - 0 = l - pos_b - 1$, $P^l_{pos_a} = P^l_{p-pos_b-1}$. This will break the Property 3.3.

If we exchange the $2i$ and $2i + 1$ dimensions of RSPE in Eq. (4), and do the addition of SPE and RSPE, the resulted additional sinusoidal positional encoding (ASPE) follows Property 3.3 for uniqueness, but it is inconsistent with Definition 3.1 and Definition 3.2. Therefore, there is no guarantee on the positional information in SBRS according to Definition 3.3.

An interesting property about this ASPE is that the pattern of uniqueness is insufficient so that the attention model cannot easily infer the positions but only to memorize. In SPE and RSPE, there is a linear combination property between two positions [37, 42]. But the ASPE breaks this property, which leads to the attention model cannot learn the relationship between different positions. We prove this difference in Appendix A.

In the literature of computer vision that utilizes attention mechanism, there is a type of encoding for images called 2D sinusoidal positional encoding (2DSPE) [2, 5, 21, 27, 48]. If pos and l are considered as the height and width, then 2DSPE is similar to the ASPE that the encoding of each pair (pos, l) is totally unique and thus, it does not follow the *forward-aware* and *backward-aware* requirements. Therefore, they are not eligible for SBRS. Detail of an example of 2DSPE is presented in Appendix B.

4 BUILDING POSITIONAL RECOMMENDER MODEL

In this section, we will derive a (learned) dual positional encoding ((L)DPE) to improve the representation ability of positional information and utilize (L)DPE to develop our Positional Recommender model for session-based recommendation.

4.1 Problem Definition

In SBRS, an item is denoted as v and there is a unique item set $\mathcal{V} = \{v_1, v_2, v_3, \ldots, v_m\}$, with m being the number of items. A session sequence from an anonymous user is defined as an order list $\mathcal{S} = [v_{s,1}, v_{s,2}, v_{s,3}, \ldots, v_{s,l}], v_{s,*} \in \mathcal{V}$. l is the length of the session $\mathcal{S}$. In this paper, a sequence has at least one item and $l \in \mathbb{Z}^+$. The goal of our model is to take an anonymous session $\mathcal{S}$ as input, and predict the next item $v_{s,l+1}$ that matches the current preference.

4.2 Dual Positional Encoding

We propose a dual positional encoding (DPE) by concatenating of half of the SPE and half of the RSPE positional encoding. The DPE is defined as:

$$
\begin{aligned}
P^l_{pos,2i} &= \sin(pos/f(i)), \\
P^l_{pos,2i+1} &= \cos(pos/f(i)), \\
P^l_{pos,2i+d/2} &= \sin((l - pos - 1)/f(i)), \\
P^l_{pos,2i+1+d/2} &= \cos((l - pos - 1)/f(i)),
\end{aligned}
\tag{5}
$$

where $i \in \{0, 1, \ldots, d/4\}$ and for clarity, we assume $d/4 \in \mathbb{Z}$ and all our results can be easily generalized to other cases.

THEOREM 4.1. *Dual positional encoding can represent the positional information of SBRS because it is both forward-aware and backward-aware.*

PROOF. (1) $\forall p, q \in \mathbb{Z}^+, \forall pos \leq \min(p, q), \exists A = \{0, 1, \ldots, d/2 - 1\}$, s.t. $P^p_{pos,A} = P^q_{pos,A}$. This follows Definition 3.1 and DPE is *forward-aware*. (2) $\forall p, q \in \mathbb{Z}^+, \forall pos_a, pos_b = \{a, b | p - a = q - b, 0 \leq a < p, 0 \leq b < q\}, \exists B = \{d/2, d/2 + 1, \ldots, d - 1\}$, s.t. $P^p_{pos_a,B} = P^q_{pos_b,B}$. This follows Definition 3.2 and DPE is *backward-aware*. Therefore, DPE can represent the positional information of SBRS.   □

This theorem states that the proposed DPE can represent the positional information of SBRS. For the $A = \{0, 1, \ldots, d/2 - 1\}$ dimensions of DPE, it is *forward-aware*. While for the rest $B = \{d/2, d/2 + 1, \ldots d - 1\}$ dimensions, it is *backward-aware*. For example, for the first item of any session with length p and q, $P_{0,A}^p = P_{0,A}^q$ is always true while no guarantee that $P_{0,B}^p = P_{0,B}^q$. But $P_{p-1,B}^p = P_{q-1,B}^q$ is always true for DPE.

The proposed DPE has met the requirements of SBRS. This encoding scheme is parameter-free. In this situation, positions in a session can be considered linear because they are ordered with a consistent interval. However, in the real world, the position of each item actually comes from the timestamp of the interaction. The time intervals between interactions are neither consistent nor linear. Therefore, we propose the following learned dual positional encoding (LDPE) to improve the inductive bias injected into a session-based recommendation model:

$$
\begin{aligned}
P_{pos,0:d/2-1}^l &= \text{Embed}[pos], \\
P_{pos,d/2:d-1}^l &= \text{Embed}[l - pos - 1],
\end{aligned}
\tag{6}
$$

where $\text{Embed} \in \mathbb{R}^{max(l) \times d}$ stands for a learned embedding matrix and $[\cdot]$ is the same as the slice operation for a list in Python. Similar to DPE, LDPE follows the Definition 3.1 and 3.2, and thus 3.3.

4.3 Positional Recommender Model

With DPE and LDPE ((L)DPE), we now build our Positional Recommender model (PosRec) based on the Position-aware Gated Graph Neural Network (PGGNN) and the bidirectional Transformer. Similar to GNN-based models [30, 49, 51], a session is firstly converted into a weighted and directed session graph. Then PGGNN is applied to calculate the position-aware item embedding as the input of the bidirectional Transformer layer. (L)DPE is incorporated into the bidirectional Transformer layer to enhance the positional information. In the end, a single vector is computed as the representation of the session and used to predict the user's next click.

4.3.1 Session Graph. To utilize the neighboring information, a session is converted into a weighted directed session graph. Similar to [30, 49, 51], the conversion procedure basically abides by the following process. If an item $v_{s,t}$ is immediately followed by the next item $v_{s,t+1}$ in the session S, then a directed edge $(w_{s,t,t+1}, v_{s,t}, v_{s,t+1})$ from this item to the next item with the edge weight $w_{s,t,t+1}$ indicating the frequency of occurrence of such an edge in S, is added to the session graph $G_s(V_s, E_s)$. V_s includes all items in the session S, and we refer to an item as a node in the following without specific indication. Each node feature $\mathbf{x}_{s,t} \in \mathbb{R}^{1 \times d}$ is initialized by the corresponding ID of the item $v_{s,t}$ and a lookup embedding matrix. E_s stands for all generated edges.

4.3.2 Position-aware Gated Graph Neural Network. The Position-aware Gated Graph Neural Network (PGGNN) is designed to process the session graph to obtain the updated item embedding. PGGNN consists of two node aggregation steps, a neighboring node aggregation based on Gated Graph Neural Network (GGNN) [25] and an anchor node aggregation based on position-aware Graph Neural Network (PGNN) [53].

GGNN for the weighted and directed session graph is defined as:

$$
\hat{\mathbf{x}_{t'}} = \sum_{v_{t'} \in N_{\text{in}}(v_t)} w_{t',t} \mathbf{x}_{t'} \mathbf{W}_{\text{in}} || \sum_{v_{t'} \in N_{\text{out}}(v_t)} w_{t,t'} \mathbf{x}_{t'} \mathbf{W}_{\text{out}},
\tag{7}
$$

$$
\mathbf{x}_t' = \text{GRU}(\mathbf{x}_t, \hat{\mathbf{x}_{t'}}),
\tag{8}
$$

where $\hat{\mathbf{x}_{t'}} \in \mathbb{R}^{1 \times 2d}$ is the message from all neighbors of v_t, $N_{\text{in}}(v_t)$ is the set of nodes targeting at v_t, $N_{\text{out}}(v_t)$ is the set of nodes targeted by v_t, $\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{out}} \in \mathbb{R}^{d \times d}$ are trainable weights and $||$ stands for the concatenation along the feature dimension. $\mathbf{x}_t'$ is the updated node feature of v_t.

Fig. 2. The pipeline of PosRec. A session S is firstly converted into a session graph G_s. PGGNN aggregates neighboring (solid edges) and anchor (dashed edges) nodes to update node features in G_s. Bidirectional Transformer uses updated node features and (L)DPE to compute a session representation $\mathbf{h}$. $\oplus$ stands for element-wise addition.

The anchor node is first introduced in PGNN by [53] with random sampling on all nodes in a graph. In the context of the session graph, nodes with notable importance, e.g., the first item, the last item and re-appearing items, can be chosen as anchor nodes. Therefore, to improve the inductive bias in the GGNN, for each node v_t in a session, we add the first item v_0 to $N_{\text{in}}(v_t)$ and the last item v_{l-1} to $N_{\text{out}}(v_t)$. In addition, for all items v_* appearing more than once in a session, v_* are added to both $N_{\text{in}}(v_t)$ and $N_{\text{out}}(v_t)$. Therefore, we substitute $N_{\text{in}}(v_t)$ and $N_{\text{out}}(v_t)$ in Eq. (7) with $N'_{\text{in}}(v_t) = \{N_{\text{in}}(v_t) + v_0 + v_*\}$ and $N'_{\text{out}}(v_t) = \{N_{\text{out}}(v_t) + v_{l-1} + v_*\}$. The corresponding weights between any anchor node and other nodes are defined as the distance of these two nodes on an unweighted and undirected graph converted from G_s by omitting the weight and direction associated with every edge. Examples of these edges are shown as dashed edges in Fig. 2.

4.3.3 Bidirectional Transformer Readout Function with (L)DPE. With the updated item features and (L)DPE, the bidirectional Transformer layer serves as the readout function to generate a feature vector for the session. The bidirectional Transformer layer operates at the graph level rather than the sequence level of the session, which will lower the noisy signal of repetitive items. Consider $\mathbf{X}' \in \mathbb{R}^{n \times d}$, which includes all updated node features, where n is the number of unique items in the session. Let $\mathbf{P}^{n \times d}$ represent (L)DPE of corresponding items of nodes in the session. The feature vector $\mathbf{h} \in \mathbb{R}^{1 \times d}$ representing the session can be defined as:

$$\mathbf{H} = \text{Transformer}(\mathbf{X}' + \mathbf{P}), \tag{9}$$

$$\mathbf{h} = \lambda_0 \mathbf{X}'_{l-1} + \lambda_1 \mathbf{H}_{l-1} + \lambda_2 \mathbf{H}_0, \tag{10}$$

where $\mathbf{H} \in \mathbb{R}^{n \times d}$ carries all output states of the bidirectional Transformer and subscripts $l-1$ and 0 represent the corresponding entries in $\mathbf{X}'$ and $\mathbf{H}$ of items v_0 and v_{l-1}. λ_* are pre-defined weights.

Note that, if a session does not contain duplicated items, every interaction will have a unique (L)DPE. When there are repeated items, the node for such an item will adopt the *forward-aware* part of (L)DPE of the earliest appearance and the *backward-aware* part of (L)DPE of the latest appearance.

4.3.4 Objective Function. After having a representation $\mathbf{h}$ of a session, we can compare $\mathbf{h}$ with the whole item set to decide what to recommend to the user. Let $\mathbf{X} \in \mathbb{R}^{m \times d}$ be the initial embedding of the whole item set. The score of recommendation $\hat{\mathbf{y}} \in \mathbb{R}^{m \times 1}$ and the Cross-Entropy loss function are defined as:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{X}\mathbf{h}^\top), \tag{11}$$

$$L = -\sum_{i=1}^{M} \mathbf{y}_i^\top \log(\hat{\mathbf{y}}_i), \tag{12}$$

where $\mathbf{y}_i \in \mathbb{R}^{m \times 1}$ is the one-hot label of training sample data i and M is the batch size.

4.4 Discussion

We will discuss the relationship and difference between the proposed PosRec model and existing recommendation methods as well as the position encoding schemes.

4.4.1 Comparisons with Existing Recommendation Methods. The proposed PosRec model makes use of a newly designed GNN module for the item representation learning and a Transformer module equipped with the (L)DPE for the session representation learning. Compared with previous GNN based methods, e.g., SR-GNN [49], GC-SAN [51], FGNN [29, 30], and MGNN-SPred [47], PosRec has a newly designed GNN module, PGGNN, which is position aware. The position awareness is the main difference between the GNN modules. In these previous work, the main contributions are including the direction information and the behavioral information in the edges. PGGNN introduces the position information in the edges, which approaches the item representation learning from a novel perspective. In addition, as these research work indicates, there is a sparsity issue in the graph construction and propagation for sessions since there could be no repeated items in the same session to construct a graph rather than a simple link list of items. Different methods try to add more connections by self-loops [30] and using the cross-session information [29]. In the proposed PGGNN method, the anchor nodes are chosen to be the first and the last item, which develops extra meaningful edges due to the requirement of calculating the relationship between normal nodes and the anchor nodes.

4.4.2 Comparisons with Existing Position Encoding Schemes. The proposed PosRec model incorporates both the *forward* and *backward* positional information with the (L)DPE. For existing recommendation models [17, 38], the most popular position encoding scheme is the LPE introduced in Attention [42]. As analyzed above, LPE is a *forward-aware* position encoding scheme. These research work has also investigated the performance of SPE, which is shown to be inferior to the LPE. It could be due to the SPE can only represent the initial intention, a less important factor compared with the latest preference. While LPE still has a possibility of learning an implicit positional information with the learnable embeddings. There are other methods to incorporate the positional information in recommendation, for example, NARM [22], STAMP [26], SR-GNN [49], and GC-SAN [51] have an attention module using the last item as the query to all other items in the session to emphasize on the last position. Such a strategy for the positional information can only consider the positional information of the last interaction, and neglect the positional information of all other interactions. While for the GNN-based methods [29, 30, 47, 49, 51], the relative positional information is contained in the direction of edges. But the relative positional information cannot reflect the absolute positional information, e.g., the latest preference and the initial intention. The proposed (L)DPE simultaneously provides the property of being both *forward* and *backward-aware*, and the learnability for a more representative embedding scheme.

5 EXPERIMENTS

In this section, we present how extensive experiments are conducted to evaluate the effectiveness of our proposed PosRec model, (L)DPE and the PGGNN module. We will answer the following research questions:

- **RQ1**: How does PosRec perform in the session-based recommendation task? (Section 5.2)
- **RQ2**: How does (L)DPE perform compared with other positional encoding schemes? (Section 5.3)
- **RQ3**: Does anchor node aggregation in PGGNN improve the recommendation? (Section 5.4)

Table 1. Statistic of datasets. (Yoo. is short for Yoochoose.)

Dataset	Clicks	♯ Train	♯ Test	Items	Avg. length
Yoo. 1/64	557248	369859	55898	16766	6.16
Yoo. 1/4	8326407	5917746	55898	29618	5.71
Diginetica	982961	719470	60858	43097	5.12

- **RQ4**: What is the visualization of DPE? (Section 5.5)
- **RQ5**: How sensitive is PosRec w.r.t. the hyper-parameters? (Section 5.6)

5.1 Setup

In this section, we will describe the experimental setup in terms of datasets (Section 5.1.1), the preprocessing procedure (Section 5.1.2), baselines (Section 5.1.3), evaluation metrics (Section 5.1.4) and the implementation (Section 5.1.5).

5.1.1 Dataset. Experiments are conducted on two benchmark datasets *Yoochoose* and *Diginetica*, which is consistent with previous methods [22, 26, 30, 49].

- *Yoochoose* is used as a challenge dataset for RecSys Challenge 2015. It is obtained by recording click-streams from an e-commerce website within 6 months. Since *Yoochoose* is a huge dataset, we follow previous methods [22, 26, 30, 49] to further divide this dataset into two subsets according to the timestamp. *Yoo. 1/64* stands for the most recent $\frac{1}{64}$ of the whole dataset and *Yoo. 1/4* for $\frac{1}{4}$ correspondingly.
- *Diginetica* is used as a challenge dataset for CIKM cup 2016. It contains the transaction data which is suitable for session-based recommendation.

The detailed statistics of each dataset can be found in Table 1.

5.1.2 Preprocessing. For the fairness and the convenience of comparison, we follow [22, 26, 30, 49] to filter out sessions of length 1 and items which occur less than 5 times in each dataset respectively. After the preprocessing step, there are 7,981,580 sessions and 37,483 items remaining in *Yoochoose* dataset, while 204,771 sessions and 43097 items in *Diginetica* dataset. Similar to [40], we split a session of length n into $n-1$ partial sessions of length ranging from 2 to n to augment the datasets. For the partial session of length i in the session S, it is defined as $[v_{s,0}, \ldots, v_{s,i-1}]$ with the last item $v_{s,i-1}$ as v_{label}. Following [22, 26, 30, 49], for *Yoochoose* dataset, the most recent portions 1/64 and 1/4 of the training sequence are used as two split datasets respectively.

5.1.3 Baselines. In order to demonstrate the advantage of the proposed PosRec model, we compare it with the following representative methods:

- **POP** is a popularity-based method that always recommends the most popular items in the whole training set, which serves as a strong baseline in some situations although it is simple.
- **S-POP** is a popularity-based method that always recommends the most popular items for the individual session.
- **Item-KNN** [34] computes the similarity of items by the cosine distance of two item vectors in sessions. Regularization is also introduced to avoid the rare high similarities for unvisited items.
- **BPR-MF** [32] proposes a BPR objective function which utilizes a pairwise ranking loss to train the ranking model. Following [22], Matrix Factorization is modified to session-based recommendation by using mean latent vectors of items in a session.

- **FPMC** [33] is a hybrid model for the next-basket recommendation and it achieves state-of-the-art results. For anonymous session-based recommendation, following [22], we omit the user feature directly because of the unavailability.
- **GRU4REC** [14] stacks multiple GRU layers to encode the session sequence into a final state. It also applies a ranking loss to train the model.
- **NARM** [22] extends to use an attention layer to combine all of the encoded states of RNN, which enables the model to explicitly emphasize on the more important parts of the input.
- **STAMP** [26] uses attention layers to replace all RNN encoders in previous work to even make the model more powerful by fully relying on the self-attention of the last item in a sequence. STAMP does not use any kind of positional encoding.
- **SR-GNN** [49] applies a gated graph convolutional layer [25] to obtain item embeddings, followed by a self-attention of the last item as STAMP does to compute the sequence level embeddings.
- **FGNN** [30] is also a graph-based recommender system, which uses the attention mechanism [42] in both the item representation learning and the item order learning.
- **GC-SAN** [51] substitutes the simple attention in the graph embedding learning of SR-GNN with multi-layer Transformers [42].

Although SASRec [17] is originally used in the sequential recommendation task rather than the SBRS task, we can still make this state-of-the-art method adapt to our experiment.

- **SASRec** is highly similar to STAMP that stacks attention layers and use the last hidden layer to predict a user's preference. SASRec makes use of LPE in its original model.

5.1.4 Evaluation metrics. For each time step, a recommender system should give out a full ranking over the whole item set. According to [20], such a ranking result will lead to a fairer comparison than sampling-based ranking methods for different models. Additionally to keep the same setting as previous baselines, we mainly choose to use two metrics, Recall and Mean Reciprocal Ranking. For both of them, we use top-5 and top-10 result to make comparisons.

- **R@K** (Recall calculated over top-K items). The R@K score is the metric that calculates the proportion of test cases which recommends the correct items in a top K position in a ranking list,

$$\text{R@K} = \frac{n_{hit}}{N}, \tag{13}$$

where N represents the number of test sequences S_{test} in the dataset and n_{hit} counts the number that the desired items are in the top K position in the ranking list, which is named the *hit*. R@K is also known as the hit ratio.

- **M@K** (Mean Reciprocal Rank calculated over top-K items). The reciprocal is set to 0 when the desired items are not in the top K position and the calculation is as follows,

$$\text{M@K} = \frac{1}{N} \sum_{v_{label} \in S_{test}} \frac{1}{Rank(v_{label})}. \tag{14}$$

The Mean Reciprocal Rank is a normalized ranking of *hit*, the higher the score, the better the quality of the recommendation because it indicates a higher ranking position of the desired item.

5.1.5 Implementation. We apply one layer of PGGNN and one layer of the attention module for our PosRec. Unless indicated otherwise, we use Adam [18] to train our model with an initial learning rate 0.001 that decreases at the rate 0.1 for every 3 epochs. The batch size and the embedding size are set to 100. To reduce the overfitting, we apply an l_2 regularization for all parameters and early stop at the end of the 4-th epoch. For weights in Eq. (10), λ_0 and λ_1 are both set to 1. For *Yoochoose*,

Table 2. Overall performance.

Method	Yoo. 1/64			
	R@5	R@10	M@5	M@10
POP	2.37	4.56	0.56	1.13
S-POP	9.96	20.18	15.25	17.96
Item-KNN	28.35±0.13	41.82±0.08	19.37±0.12	21.24±0.07
BPR-MF	7.64±0.18	20.47±0.14	8.58±0.15	11.65±0.11
FPMC	22.93±0.09	35.38±0.19	11.83±0.17	14.57±0.10
GRU4REC	37.81±0.08	50.30±0.13	20.13±0.09	22.81±0.05
NARM	44.69±0.12	57.56±0.09	25.43±0.05	27.16±0.08
STAMP	46.42±0.13	58.67±0.07	28.05±0.08	29.66±0.10
SR-GNN	47.33±0.07	60.04±0.11	28.32±0.10	30.08±0.12
FGNN	47.12±0.10	60.13±0.08	28.45±0.06	30.17±0.09
GC-SAN	46.48±0.08	59.47±0.11	27.58±0.18	29.33±0.06
SASRec	46.65±0.16	58.98±0.10	28.13±0.12	29.87±0.13
PosRec	**47.96**±0.15	**60.90**±0.07	**28.83**±0.12	**30.57**±0.09

Method	Yoo. 1/4			
	R@5	R@10	M@5	M@10
POP	0.76	0.98	0.09	0.15
S-POP	8.69	18.57	14.84	16.87
Item-KNN	30.16±0.15	41.86±0.13	18.54±0.09	20.29±0.14
BPR-MF	1.15±0.11	2.61±0.06	0.79±0.05	1.13±0.07
FPMC	-	-	-	-
GRU4REC	36.80±0.10	49.60±0.12	19.71±0.08	21.43±0.05
NARM	44.95±0.08	57.73±0.07	25.60±0.09	27.39±0.11
STAMP	45.38±0.12	58.03±0.10	27.38±0.12	29.08±0.09
SR-GNN	47.71±0.09	60.64±0.09	28.33±0.07	30.24±0.06
FGNN	47.63±0.14	60.68±0.13	28.43±0.10	30.19±0.08
GC-SAN	46.97±0.07	59.86±0.14	28.12±0.08	29.72±0.09
SASRec	45.21±0.18	57.88±0.108	27.46±0.17	29.23±0.15
PosRec	**47.97**±0.12	**60.92**±0.18	**29.29**±0.07	**31.03**±0.08

Method	Diginetica			
	R@5	R@10	M@5	M@10
POP	0.34	0.68	0.06	0.13
S-POP	2.67	9.95	9.32	11.79
Item-KNN	11.75±0.09	24.09±0.08	8.34±0.12	10.63±0.18
BPR-MF	0.78±0.10	2.28±0.14	0.07±0.03	0.83±0.06
FPMC	4.49±0.07	15.71±0.012	2.86±0.21	5.17±0.13
GRU4REC	21.87±0.06	32.67±0.12	11.83±0.09	13.26±0.07
NARM	24.43±0.07	36.09±0.09	12.48±0.07	14.02±0.10
STAMP	25.85±0.07	37.46±0.10	14.42±0.08	15.96±0.12
SR-GNN	26.53±0.09	37.87±0.14	14.99±0.05	16.49±0.06
FGNN	26.46±0.13	37.82±0.08	15.06±0.12	16.55±0.13

GC-SAN	26.29±0.08	37.75±0.07	14.73±0.12	16.23±0.09
SASRec	25.87±0.13	37.53±0.09	14.37±0.16	16.12±0.11
PosRec	**27.30**±0.12	**38.87**±0.16	**15.31**±0.09	**16.85**±0.08

λ_2 is set to 0.5 while for *Diginetica*, it is set to 1. And the default position encoding is set to LDPE if there is no further indication. We use one Nvidia GeForce RTX 2080 ti GPU for training.

5.2 Overall Performance

The overall recommendation performance is demonstrated in Table 2. We compare the PosRec with the following baselines: (1) shallow methods: POP, S-POP, Item-KNN [34], BPR-MF [32] and FPMC [33]; (2) GRU-based methods: GRU4REC [14] and NARM [22]; (3) Attention-based method: STAMP [26]; (4) GNN-based methods: SR-GNN [49], FGNN [30] and GC-SAN [51] and (5) adapted sequential methods: SASRec [17].

Our PosRec achieves the best performance compared with all baselines across all the datasets by four metrics. Compared with the previous state-of-the-art methods, the improvement is consistent. Our proposed PosRec method effectively exploits the positional information in the positional encoding module and the PGGNN module.

The traditional methods generally cannot compete with the current deep learning models. The popularity-based methods, POP and S-POP, simply recommend items based on the frequencies of appearance in the whole dataset and the current session respectively. These popularity-based approaches tend to recommend fixed items in a general situation, which is not able to learn to recommend the proper items. Although the S-POP model can make use of the session information to improve the performance compared with the POP, it still lacks the ability to learn the session pattern. PosRec performs much better compared with both of them. For shallow learning-based methods, BPR-MF and FPMC, they do not consider the session information in the recommendation, they do not have a competitive performance neither scale up well in the larger dataset *Yoochoose* 1/4. Item-KNN outperforms all the traditional methods. Item-KNN only considers the closeness between items while avoiding sequential information. Generally, these traditional methods cannot compete with the neural network-based recommender systems.

GRU4REC is the first session-based model that uses a recurrent structure to capture sequential information. GRU4REC outperforms the shallow and popularity-based methods by a large margin, which can be viewed as a strong baseline. The sequential information is encoded explicitly by the recurrent calculation procedure. But the positional information of a session is only included implicitly along with the recurrent calculation. A big issue of the recurrent-based methods is that there is a catastrophic forgetting of the early information. For the attention-based baselines, NARM and STAMP, they apply the self-attention mechanism mainly over the last item, which considers the last item as the pivot item to represent the latest intention. This structure is a better inductive bias in positional information than linearly rolling out as RNN structure by outperforming the GRU4REC. This situation is also proved by the better performance of STAMP compared with NARM. STAMP completely excludes the recurrent structure and relies heavily on the last item. SASRec achieves a slightly higher result than STAMP since they both perform attention while SASRec includes a learned absolute positional encoding. Recent GNN-based methods, SR-GNN, FGNN and GC-SAN, have made improvements by utilizing the connectivity of items. These methods use a session graph to represent the session sequence by linking the interactions according to their chronological order. The relative positional information between interactions is included in the edge and in the graph readout calculation step, it resembles the attention mechanism to focus more on the last item. The

Fig. 3. Performance of different positional encoding schemes on *Yoo. 1/64*.

Fig. 4. Performance of different positional encoding schemes on *Diginetica*.

positional information is only partially considered with the relative positional information and the *backward-awareness*.

5.3 Different Positional Encoding Schemes

To evaluate the effect of (L)DPE, we substitute (L)DPE with the following encoding schemes in the bidirectional Transformer module in PosRec: SPE (sinusoidal positional encoding), LPE (learned positional encoding), RSPE (relative sinusoidal positional encoding), LRPE (learned relative positional encoding), ASPE (additional sinusoidal positional encoding), ALPE (additional learned positional encoding), 2DSPE (2D sinusoidal positional encoding) and 2DLPE (2D learned positional encoding), where S indicates a sinusoidal scheme and L indicates a learned scheme. The result is presented in Fig. 3 and 4. To further verify the ubiquitous efficacy of LDPE, we integrate LDPE into the attention-based models, STAMP and SASRec. The result is shown in Table 3.

Among Fig. 3 and 4, the blue line in each chart indicates the base model that does not include any positional encoding in the PosRec model. Compared with other positional encoding schemes, LDPE can consistently improve the recommendation performance in all situations. For the learnable scheme, LPE and LRPE both achieve relatively good performance because they represent parts of the positional information. The LPE scheme is adopted by SASRec [17] and BERT4Rec [38] for the sequential recommendation as well. There is a small gap for LPE to outperform LRPE. It could be because LPE provides a further *forward-aware* information while the readout function has already included the last item as the *backward-aware* information. ALPE and 2DLPE are not theoretically considered as suitable for SBRS. But they still have a comparable result because their entries of the PE are unique so that the model can still learn the pattern of the encoding but in a harder way. As for the parameter-free scheme, LPE and RSPE achieve comparable results with DPE because they provide reasonable positional information to the model. ASPE has the worst result because the *forward-aware* and *backward-aware* information is entangled in the representation. In contrast, 2DSPE has a clearer connection between entries than ASPE.

In Table 3, it is shown that LDPE can consistently improve the performance of attention-based models. STAMP does not originally use any positional encoding. While SASRec has already utilized

Method	Yoo. 1/64				Diginetica			
	R@5	R@10	M@5	M@10	R@5	R@10	M@5	M@10
STAMP	46.42	58.67	28.05	29.66	25.85	37.46	14.42	15.96
STAMP+LDPE	46.78	58.82	28.31	29.94	26.23	37.81	14.79	16.34
SASRec	46.65	58.98	28.13	29.87	25.87	37.53	14.37	16.12
SASRec+LDPE	46.89	59.17	28.37	30.11	26.26	37.90	14.71	16.53

Table 3. Performance of LDPE on attention-based models.

Fig. 5. Performance of anchor node aggregation on *Yoo. 1/64*.

Fig. 6. Performance of anchor node aggregation on *Diginetica*.

an LPE in its basic method. For both *Yoochoose* and *Diginetica* dataset, LDPE can improve the performance of LPE by a large margin with the same number of parameters, which verifies that LDPE can be integrated into different models.

5.4 Effect of Anchor Node Aggregation

To evaluate the effect of the anchor node aggregation in PGGNN, we add this module to GNN-based methods: SR-GNN, FGNN, GC-SAN and PosRec-ANC (indicating the PosRec model removing the anchor node aggregation). The result is presented in Fig. 5 an 6.

It is shown that with the anchor node aggregation, the performance of all GNN-based methods has an increase for Recall. While for the MRR metric, only GC-SAN and PosRec gain a steady improvement. Anchor nodes improve the model performance: (1) injecting the inductive bias of the distance between normal items and important items to the model; (2) increasing the connectivity of the session graph. On one hand, the *position-aware* node feature learning is proved by [53] that it can provide the positional information in addition to the structure information. On the other hand, session data is originally sparse in the aspect of connectivity. A large portion of session data

(a) SPE of len. 10.   (b) DPE of len. 10.

Fig. 7. Positional encoding visualization for the session length of 10.

(a) SPE of len. 20.   (b) DPE of len. 20.

Fig. 8. Positional encoding visualization for the session length of 20.

(a) SPE of len. 10.   (b) SPE of len. 20.   (c) DPE of len. 10.   (d) DPE of len. 20.

Fig. 9. Nearest neighbor heatmap of the same length.

is even a simple sequence without any repetitive items. After including the aggregation of anchor nodes, the connectivity is increased because there are additional edges between normal nodes and anchor nodes. GNN layers are more suitable to process such data.

5.5 Visualization

In this experiment, we visualize how SPE, DPE, LPE, and LDPE look like and their characteristics. We firstly show the SPE and DPE themselves in Fig. 7. We examine the nearest neighbor relationship between encoding of different positions in Fig. 9. Example session lengths are set to 10 and 20 and the embedding size is set to 100. Fig. 7(a) and 7(b) show the SPE and DPE for length 10 respectively. We further present the heatmap of LPE and LDPE of length 10 and 20 in Fig. 11 and Fig. 12, which are learned based on the experiment on *Yoo. 1/64*.

The nearest neighbor heatmaps shown in Fig. 9 represent the dot product of the positional encoding of the same session length. SPE and DPE of lengths 10 and 20 are demonstrated respectively. By definition, the heatmap of RSPE is the same as SPE. We can see that within the same length, these positional encoding schemes do have a strong correlation to the same position of the same length.

To examine the correlation of the positional encoding from different session lengths, we demonstrate the results from the dot product between lengths of 10 and 20 in Fig. 10. For the case of SPE in Fig. 10(a),there is one diagonal strip that a strong correlation is only in between the same

(a) SPE of len. 10 and 20.   (b) DPE of len. 10 and 20.

Fig. 10. Nearest neighbor heatmap of length 10 and 20.

(a) LPE of len. 10.   (b) LPE of len. 20.   (c) LPE of len. 10 and 20.

Fig. 11. Nearest neighbor heatmap of length 10 and 20 of LPE.

(a) LDPE of len. 10.   (b) LDPE of len. 20.   (c) LDPE of len. 10 and 20.

Fig. 12. Nearest neighbor heatmap of length 10 and 20 of LDPE.

position but not the same reverse position. For example, there is no strong correlation between the last position of length 10 and length 20 (0.16 in the lower right corner). While for the case of DPE in Fig. 10(b), two diagonal strips indicate a strong correlations in both direction of positions.

From Fig. 7(a) and 8(a), it is clear to see the *forward-awareness* of SPE. For example, no matter what length is, the encoding for the first position will always be the same (the first row). While for the last position of lengths 10 and 20, as the max length and the embedding size vary, there will not be a shared same part, which is not *backward-aware*. For RSPE, the case is just the reverse, i.e., the heat map would be upside down of SPE. From Fig. 7(b) and 8(b), both *forward-awareness* and *backward-awareness* are demonstrated. The same positions and reverse positions will always share the same half of the embedding respectively.

The nearest neighbor heatmaps of LPE and LDPE are shown in Fig. 11 and Fig. 12 respectively. From the heatmaps between the same length for 10 and 20 in Fig. 11(a) and Fig. 11(b), the LPE

(a) *Yoo. 1/64.*   (b) *Diginetica.*

Fig. 13. Sensitivity of λ_2.

focuses mostly on the initial intention with the most related encoding being the first position. From the heatmap between the length 10 and 20 in Fig. 11(c), it is obvious that LPE can only capture the *forward* positional information of the initial intention of sessions with different lengths. The position encodings close to the end of sessions are not carrying *forward* or *backward* positional information, which is shown at the bottom right corner of Fig. 11(c) with a similar closeness between different positions. Compared with LPE, the heatmaps of LDPE are shown in Fig. 12(a) and Fig. 12(b), from which the diagonal cells indicate that the LDPE can capture the positional information of distinct positions of the same length. While for Fig. 12(c), it can be seen that the LDPE can capture the *backward* positional information with the light cell at the bottom right corner. For the *forward* positional information, the heatmap at the upper left corner shows a diagonal pattern. While the diagonal pattern is more obvious on the right part in Fig. 12(c), which indicates that the *backward* positional information is more important and useful for SBRS.

5.6 Parameter Sensitivity

In this experiment, we evaluate the effect of λ_0, λ_1 and λ_2 in Eq. 10. We evaluate the relative scale between these three hyper-parameters. We fix λ_0 and λ_1 to 1 because they both represent the strength about the information from the last position. And we change λ_2 from $\{0.25, 0.5, 1, 2\}$. In Fig. 13, we demonstrate the result of the sensitivity of λ_2 is shown. For the *Yoochoose* dataset in Fig. 13(a), the best performance is achieved when $\lambda_2 = 0.5$. Compared with $\lambda_2 = 0.25$, we can see that inadequate initial intent will do harm to the overall performance. When it comes to $\lambda_2 = 1$ or 2, we can also imply that the initial intent is not as important as the latest preference. For the *Diginetica* dataset in Fig. 13(b), the best performance is achieved when $\lambda_2 = 1$. Compared with $\lambda_2 = 0.25$ or 0.5, we can see that more initial intent will benefit the overall performance.

6 CONCLUSION

In this paper, we investigate how the positional information can be exploited in the session-based recommendation. We find that there are two types of positional information, *forward* and the *backward* to represent the initial and the latest intentions in a session. A theoretical framework is proposed to analyze the representation ability of positional encoding schemes for the positional information in the session-based recommendation task. Specifically, the *forward-awareness* and the *backward-awareness* are defined for the evaluation of these schemes. Conventionally, a positional encoding is designed for language models with only the *forward-awareness*. However, session-based recommendation requires both of the the *forward-awareness* and the *backward-awareness*. Therefore, a novel (learned) dual positional encoding scheme ((L)DPE) is proposed to fully capture both of the *forward* and the *backward* positional information. Besides, we design a PGGNN module to enhance the representation ability for graph neural networks to make use of the positional information. Combining both (L)DPE and the PGGNN, we build a PosRec model to perform the

session-based recommendation with effective exploitation of the positional information. Extensive experiments are conducted on two benchmark datasets, which demonstrate that our proposal can achieve state-of-the-art performance and effectively exploit the positional information in SBRS.

REFERENCES

[1] Mohammad Aliannejadi and Fabio Crestani. 2018. Personalized Context-Aware Point of Interest Recommendation. *ACM Trans. Inf. Syst.* 36 (2018).

[2] Irwan Bello, Barret Zoph, Quoc Le, Ashish Vaswani, and Jonathon Shlens. 2019. Attention Augmented Convolutional Networks. In *ICCV*.

[3] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *WSDM*.

[4] Yi Cao, Weifeng Zhang, Bo Song, Weike Pan, and Congfu Xu. 2020. Position-aware context attention for session-based recommendation. *Neurocomputing* 376 (2020).

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. *CoRR* abs/2005.12872 (2020).

[6] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *CoRR* abs/1905.06874 (2019).

[7] Tong Chen, Hongzhi Yin, Quoc Viet Hung Nguyen, Wen-Chih Peng, Xue Li, and Xiaofang Zhou. 2020. Sequence-Aware Factorization Machines for Temporal Predictive Analytics. In *ICDE*.

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

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *ACL*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[11] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Trans. Inf. Syst.* 39 (2020).

[12] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-based Recommendation. In *SIGKDD*.

[13] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *CIKM*.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[15] Balázs Hidasi and Domonkos Tikk. 2016. General factorization framework for context-aware recommendations. *Data Min. Knowl. Discov.* 30 (2016).

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9 (1997).

[17] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *SIGKDD*.

[21] Junyeop Lee, Sungrae Park, Jeonghun Baek, Seong Joon Oh, Seonghyeon Kim, and Hwalsuk Lee. 2019. On Recognizing Texts of Arbitrary Shapes with 2D Self-Attention. *CoRR* abs/1910.04396 (2019).

[22] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*.

[23] Yang Li, Tong Chen, Yadan Luo, Hongzhi Yin, and Zi Huang. 2021. Discovering Collaborative Signals for Next POI Recommendation with Iterative Seq2Graph Augmentation. *CoRR* abs/2106.15814 (2021).

[24] Yang Li, Yadan Luo, Zheng Zhang, Shazia W. Sadiq, and Peng Cui. 2019. Context-Aware Attention-Based Data Augmentation for POI Recommendation. In *ICDE Workshops*.

[25] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.

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

[27] Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. 2019. Stand-Alone Self-Attention in Vision Models. In *NeurIPS*.

[28] Tieyun Qian, Bei Liu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2019. Spatiotemporal Representation Learning for Translation-Based POI Recommendation. *ACM Trans. Inf. Syst.* 37 (2019).

[29] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting Cross-session Information for Session-based Recommendation with Graph Neural Networks. *ACM Trans. Inf. Syst.* 38 (2020).

[30] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *CIKM*.

[31] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation. In *SIGIR*.

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

[33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*.

[34] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

[35] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (2005).

[36] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

[37] Vighnesh Leonardo Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *NeurIPS*.

[38] 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*.

[39] Ke Sun, Tieyun Qian, Hongzhi Yin, Tong Chen, Yiqi Chen, and Ling Chen. 2019. What Can History Tell Us?. In *CIKM*.

[40] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS@RecSys*.

[41] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.

[43] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[44] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order Matters: Sequence to sequence for sets. In *ICLR*.

[45] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On Position Embeddings in BERT. In *ICLR*.

[46] Chenyang Wang, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2021. Toward Dynamic User Intention: Temporal Evolutionary Effects of Item Relations in Sequential Recommendation. *ACM Trans. Inf. Syst.* 39 (2021).

[47] Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Beyond Clicks: Modeling Multi-Relational Item Graph for Session-Based Target Behavior Prediction. In *WWW*.

[48] Zelun Wang and Jyh-Charn Liu. 2019. Translating Mathematical Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training. *CoRR* abs/1908.11415 (2019).

[49] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *AAAI*.

[50] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In *AAAI*.

[51] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*.

[52] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

[53] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. In *ICML*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.).

[54] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *WSDM*.

[55] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using Temporal Data for Making Recommendations. In *UAI*.

A PROOF OF PROPERTY OF ASPE

PROOF. From [37, 42], SPE has a linear combination property as follows:

$$P^l_{x+y,2i} = P^l_{x,2i}P^l_{y,2i+1} + P^l_{x,2i+1}P^l_{y,2i}$$
$$P^l_{x+y,2i+1} = P^l_{x,2i+1}P^l_{y,2i+1} - P^l_{x,2i}P^l_{y,2i}. \tag{15}$$

The modified RSPE is defined as:

$$P^l_{l-pos-1,2i} = \cos((l-pos-1)/10000^{2i/d})$$
$$P^l_{l-pos-1,2i+1} = \sin((l-pos-1)/10000^{2i/d}). \tag{16}$$

For simplicity, we redefine $POS = pos/10000^{2i/d}$ and $L = (l-1)/10000^{2i/d}$.
The addition of SPE and RSPE here is as follows:

$$P^l_{pos,2i} = \sin(POS) + \cos(L - POS)$$
$$P^l_{pos,2i+1} = \cos(POS) + \sin(L - POS). \tag{17}$$

For positions x and y:

$$
\begin{aligned}
P^l_{x,2i} &= \sin(X) + \cos(L - X) \\
&= (1 + \sin(L))\sin X + \cos L \cos X \\
P^l_{x,2i+1} &= \cos(X) + \sin(L - X) \\
&= (1 + \sin(L))\cos X + \cos L \sin X \\
P^l_{y,2i} &= \sin(Y) + \cos(L - Y) \\
&= (1 + \sin(L))\sin Y + \cos L \cos Y \\
P^l_{y,2i+1} &= \cos(Y) + \sin(L - Y) \\
&= (1 + \sin(L))\cos Y + \cos L \sin Y \\
P^l_{x+y,2i} &= \sin(X + Y) + \cos(L - (X + Y)) \\
&= (1 + \sin(L))\sin(X + Y) + \cos(L)\cos(X + Y) \\
P^l_{x+y,2i+1} &= \cos(X + Y) + \sin(L - (X + Y)) \\
&= (1 + \sin(L))\cos(X + Y) + \cos(L)\sin(X + Y).
\end{aligned} \tag{18}
$$

Similar to Eq. (15), we calculate the multiplication between encoding:

$$
\begin{aligned}
P^l_{x,2i}P^l_{y,2i+1} &= (1 + \sin(L))^2 \sin(X)\cos(Y) + (1 + \sin(L))\cos(L)\sin(X)\sin(Y) \\
&\quad + (1 + \sin(L))\cos(L)\cos(X)\cos(Y) + \cos^2(L)\cos(X)\sin(Y)
\end{aligned} \tag{19}
$$

$$
\begin{aligned}
P^l_{x,2i+1}P^l_{y,2i} &= (1 + \sin(L))^2 \sin(Y)\cos(X) + (1 + \sin(L))\cos(L)\cos(X)\cos(Y) \\
&\quad + (1 + \sin(L))\cos(L)\sin(X)\sin(Y) + \cos^2(L)\sin(X)\sin(Y)
\end{aligned} \tag{20}
$$

$$
\begin{aligned}
P^l_{x,2i+1}P^l_{y,2i+1} &= (1 + \sin(L))^2 \cos(X)\cos(Y) + (1 + \sin(L))\cos(L)\cos(X)\sin(Y) \\
&\quad + (1 + \sin(L))\cos(L)\sin(X)\cos(Y) + \cos^2(L)\sin(X)\sin(Y)
\end{aligned} \tag{21}
$$

$$
\begin{aligned}
P^l_{x,2i}P^l_{y,2i} &= (1 + \sin(L))^2 \sin(X)\sin(Y) + (1 + \sin(L))\cos(L)\sin(X)\cos(Y) \\
&\quad + (1 + \sin(L))\cos(L)\cos(X)\sin(Y) + \cos^2(L)\cos(X)\cos(Y)
\end{aligned} \tag{22}
$$

For simplicity, we define A = Eq. (19) + Eq. (20), B = Eq. (19) − Eq. (20), C = Eq. (21) + Eq. (22) and D = Eq. (21) − Eq. (22)

Therefore, $P^l_{x+y,2i}$ and $P^l_{x+y,2i+1}$ can be computed based on A, B, C and D:

$$
\begin{aligned}
P^l_{x+y,2i} &= \frac{A - \cos(L)C}{2(1 + \cos^2(L))} + \frac{\cos(L)D}{2\sin(L)(1 + \sin(L))} \\
P^l_{x+y,2i+1} &= \frac{D}{2\sin(L)} + \frac{\cos(L)C - A}{s(1 + \sin(L))(\cos(L) - 1)}.
\end{aligned}
\tag{23}
$$

Therefore, the relationship between the PE of two positions based on the addition of SPE and RSPE cannot be represented as linear combination because L is a variable among different sessions. □

B EXAMPLE OF 2DSPE

$$
\begin{aligned}
P^l_{pos,2i} &= \sin\left(pos/10000^{4i/d}\right) \\
P^l_{pos,2i+1} &= \cos\left(pos/10000^{4i/d}\right) \\
P^l_{pos,2i+d/2} &= \sin\left(l/10000^{4i/d}\right) \\
P^l_{pos,2i+1+d/2} &= \cos\left(l/10000^{4i/d}\right)
\end{aligned}
\tag{24}
$$