PeerJ Computer Science

A dynamic graph Hawkes process based on linear complexity self-attention for dynamic recommender systems

Zhiwen Hou[1], Xiaojun Lv[2], Yuchen Zhou[1], Lingbin Bu[1], Qiming Ma[1], Yifan Wang[1] and Fanliang Bu[1]

[1] School of Information Network Security, People's Public Security University of China, Beijing, China
[2] Institute of Computing Technology, China Academy of Railway Sciences Corporation Limited, Beijing, China

ABSTRACT

The dynamic recommender system realizes the real-time recommendation for users by learning the dynamic interest characteristics, which is especially suitable for the scenarios of rapid transfer of user interests, such as e-commerce and social media. The dynamic recommendation model mainly depends on the user-item history interaction sequence with timestamp, which contains historical records that reflect changes in the true interests of users and the popularity of items. Previous methods usually model interaction sequences to learn the dynamic embedding of users and items. However, these methods can not directly capture the excitation effects of different historical information on the evolution process of both sides of the interaction, *i.e.*, the ability of events to influence the occurrence of another event. In this work, we propose a Dynamic Graph Hawkes Process based on Linear complexity Self-Attention (DGHP-LISA) for dynamic recommender systems, which is a new framework for modeling the dynamic relationship between users and items at the same time. Specifically, DGHP-LISA is built on dynamic graph and uses Hawkes process to capture the excitation effects between events. In addition, we propose a new self-attention with linear complexity to model the time correlation of different historical events and the dynamic correlation between different update mechanisms, which drives more accurate modeling of the evolution process of both sides of the interaction. Extensive experiments on three real-world datasets show that our model achieves consistent improvements over state-of-the-art baselines.

Subjects Artificial Intelligence, Data Mining and Machine Learning, Neural Networks
Keywords Recommender systems, Dynamic graph, Hawkes process, Self-attention

Submitted 23 November 2022
Accepted 4 April 2023
Published 9 May 2023

Corresponding author
Fanliang Bu, bufanliang@sina.com

Academic editor
Dan Stowell

Additional Information and Declarations can be found on page 19

DOI **10.7717/peerj-cs.1368**

Copyright
2023 Hou et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

INTRODUCTION

Dynamic recommendation is an important application of big data methods, which has been successfully applied in many fields, such as e-commerce, social media, health and medical services. In addition, dynamic recommendation can be implemented for public safety, such as predicting security events in a specific area (*Liu et al., 2016b*; *Wu et al., 2016*; *Liu, Wu & Wang, 2017*).

The real-world dynamic recommender system predicts the item that the user may interact with in the future by mining the time series information of the interaction between

users and items. The two types of entities, users and items, can cover a variety of notions, *e.g.*, users could be customers in an ecommerce system, or accounts on social media; items could be products, posts, media produced or consumed by users. In real world scenarios, both user interest and item popularity may shift and evolve along with time. Besides, the historical interaction information (*Hidasi et al., 2015*; *Li et al., 2017*; *Liu et al., 2016a*; *Wu et al., 2017*) and the historical collaborative interaction information (*Rendle, Freudenthaler & Schmidt-Thieme, 2010*; *Koren, 2009*; *Rendle et al., 2012*) of users or items are proved powerful in making recommendation. The historical interaction information here refers to the historical interaction item (or user) sequence of the user(or item), and the historical collaborative interaction information refers to the second-order historical interaction node sequence obtained by using the high-order connectivity of the graph structure. Consequently, the key to building a dynamic recommender system is how to effectively combine the dynamic changes of users and items with their historical information.

At present, several approaches have been proposed to predict future items a user is likely to interact with, providing encouraging results (*Hidasi et al., 2015*; *Wu et al., 2017*; *Covington, Adams & Sargin, 2016*; *Dai et al., 2016*; *Kumar, Zhang & Leskovec, 2019*; *Tan, Xu & Liu, 2016*; *Wang et al., 2020*; *Wu et al., 2019b*; *Xu et al., 2019*). Often, however, the focus is on modeling users, while the historical interaction dynamics that provide a richer signal is overlooked (*Wang et al., 2019*). Therefore, recurrent neural networks (RNN) and other models suitable for sequences (*Hidasi et al., 2015*; *Li et al., 2017*; *Liu et al., 2016a*; *Wu et al., 2017*) are used to model the long-term dependencies of item sequences. Recently, some studies have shown that significant improvements to traditional methods can be achieved by building a user-item bipartite graph and modeling two types of entities at the same time (*Dai et al., 2016*; *Kumar, Zhang & Leskovec, 2019*). Unlike previous methods, JODIE (*Kumar, Zhang & Leskovec, 2019*) and other dynamic evolution models (*You et al., 2019*; *Wang et al., 2016*; *Wu et al., 2019a*) have employed mutually recursive RNNs that are more capable to model the user-item interaction dynamics. Dynamic graph collaborative filtering (DGCF) (*Li et al., 2020*) also uses the graph structure to model the collaborative information between nodes. However, these methods often ignore the influence of different historical interaction information and historical collaborative interaction information on the evolution process of the current interaction. Nonetheless, the Hawkes process can effectively capture this effect (*Hawkes, 1971*; *Mei & Eisner, 2017*), which assumes that historical events will increase the probability of future events to some extent, and this effect will decay with time. Therefore, the evolution process of user or item embedding can be effectively modeled by using this property. In this process, the users or items involved in the interaction when each link is formed can be considered to be determined by the influence of recent events. For example, in a product recommendation network, a particular product that has attracted a large number of buyers recently may attract more buyers in the near future.

Therefore, to effectively model the user or item embedding evolution process on the dynamic graph, we propose a Dynamic Graph Hawkes Process based on Linear complexity Self- Attention (DGHP-LISA) for dynamic recommender systems. First, we propose a new dynamic graph embedding update framework based on the Hawkes process, which

can not only inherits the previous embeddings during the node embedding update, but also explicitly model the excitation effects of the evolution process of both sides of the current interaction through historical interaction information and historical collaborative interaction information. In addition, we design a new self-attention mechanism with linear complexity (LISA), which is efficient enough to be the backbone component of a deep network. Therefore, we add the LISA module in the time encoding of neighbor nodes and the feature fusion of node embeddings to capture the time correlation of different historical events and the dynamic correlation between different update mechanisms respectively. We evaluate the effectiveness of DGHP-LISA on three public datasets through experiments, and the results show that DGHP-LISA achieves the most advanced performance.

RELATED WORK

Different from static recommendation models such as matrix factorization (MF) and Bayesian personalized ranking (BPR), the main task of dynamic recommendation models is to capture the dynamic changes of users and items from historical and current interactions, so as to accurately predict the embedding trajectory of users or items over time to recommend items that users may interact with in the future. In the field of dynamic recommendation, traditional methods use a vanilla deep neural network (DNN) architecture to integrate manually designed or learned features into the recommendation model. For example, *Covington, Adams & Sargin (2016)* divided the recommender system of YouTube into two stages: recall and sorting, and the two-stage model used DNN with similar structure to provide personalized recommendations for a large number of users in millions of candidate videos.

As recurring patterns in user-item interactions are considered to be critical in recommending or predicting future activities, RNNs and its variants have been widely used in interaction prediction. Specifically, *Hidasi et al. (2015)* creatively applied RNN to session-based recommendation problems. The model predicts the next item clicked by the user according to the click sequence information of the user's current session. As a variant of RNN, *Wu et al. (2017)* used long short-term memory (LSTM) to build a model to capture the dynamic changes of users and movies. *Beutel et al. (2018)* proposed the LatentCross model based on the gated recurrent unit (GRU), which incorporates multiple types of contextual information into the model by taking the dot product of the contextual features and the hidden states in the GRU.

Since activities that are close to an event in time are more likely to trigger such event than the ones that are far apart, encoding the time interval between activities is helpful for improving the performance of the recommender system. However, standard RNNs and its variants cannot handle the time intervals between historical events. Therefore, driven by the above requirements, many works have extended RNNs and their variants to fully account for time intervals (*Zhang, 2019*; *Zhu et al., 2017*). For example, *Zhu et al. (2017)* designed a time gate for LSTM, so that the model can not only effectively deal with serialized data, but also model time information very well.

However, all the above methods only simulate the changes of users' interests through the user's historical interaction sequence, while ignoring the evolution of the item. In order to

solve this problem, many works have leveraged point process models and RNNs to jointly learn the dynamic embedding of users and items. *Dai et al. (2016)* applied the temporal point process to dynamic recommendation and design learnable intensity functions to model the dynamic impact between users and items. *Kumar, Zhang & Leskovec (2019)* defined embedding update operation and embedding projection operation to learn the dynamic embedding of users and items from a series of time interactions. *Li et al. (2020)* introduced dynamic graph into dynamic recommendation task for the first time, and extended JODIE by considering the second-order neighborhood information of dynamic interaction networks.

Our DGHP-LISA is inspired by the above work. However, these latest works do not take into account the impact of different historical interaction information and historical collaborative interaction information on the evolution of both sides of the interaction. Overall, the main contributions of the proposed model to dynamic recommendation scenarios are as follows:

1. We introduce Hawkes process and dynamic graph into dynamic recommendation scenarios at the same time to model the dynamic interaction and evolution between users and items;
2. We design a new self-attention with linear complexity and introduce it into the node embedding update framework based on dynamic graph Hawkes process. On the one hand, when time-encoding neighbor nodes, the LISA module can effectively model the time decay effects of different historical events on current events. On the other hand, when fusing features generated by different update mechanisms, the LISA module can effectively model the importance between update mechanisms by dynamically allocating weights;
3. Extensive experiments have been carried out on three public datasets. The experimental results show that, compared with the most advanced baseline, our DGHP-LISA achieves the best results.

PRELIMINARIES

Dynamic graph recommendation

In this work, the goal of dynamic recommendation is to learn the representation of users and items from current interactions and history, and then predict the items that users are most likely to be interested in in the future. This ‘‘interest’’ is the relationship between the user and the recommended item. The graph is the basic data structure to express this relationship, which can well describe the relationship between users and items. Therefore, the graph structure is used to solve many problems in the field of dynamic recommendation (*Li et al., 2020*; *Trivedi et al., 2019*). Since a dynamic graph can naturally and effectively represent the interaction between users and items over time, we use a dynamic graph structure to model a dynamic recommendation problem.

Definition 1 Dynamic graph. In essence, the dynamic graph here is a bipartite graph. Both the user and the item are nodes, and all interactions are located between the user and the item node. If the user interacts with the item at the time

$t \in \mathbb{R}^+, \forall t \in [0, T]$, it forms a dynamic graph $G_t = \left(U \bigcup V, E_t\right)$. Where U and V represent the set of users and items, respectively, and $U \bigcup V$ is a finite set of all users and items. E_t is a finite set of interactions in G_t, *i.e.,* a set of all interactions between users and items before time t. The dynamic graph $G_{t0} = \left(U \bigcup V, E_{t0}\right)$ of the initial time t_0 consists of isolated nodes or snapshots of the dynamic graph, and the initial embeddings of users and items are initial feature vectors or random vectors.

Definition 2 User-item interaction events. In dynamic recommendation, the user-item interaction event is a triple (u, v, t), which represents the formation of interaction $(u, v) \in E_t$ between user node $u \in U$ and item node $v \in V$ at time t. Therefore, a dynamic graph can also be defined as a chronological sequence of user-item interaction events $I = \left\{(u, v, t)_n : n = 1, 2, \ldots, |E_T|\right\}$.

Definition 3 Static and dynamic embeddings. To encode long-term static properties and temporal dynamic characteristics of entities, all users and items are assigned static and dynamic embeddings. On the one hand, we denote the static embeddings of user u and item v at time t as $\bar{u} \in \mathbb{R}^n$ and $\bar{v} \in \mathbb{R}^n$, which do not vary with time. Following previous proposals (*Zhu et al., 2017*; *Baytas et al., 2017*), we use one-hot vectors as static embeddings for all users and items. On the other hand, we assign dynamic embedding represented by $u(t) \in \mathbb{R}^d$ and $v(t) \in \mathbb{R}^d$ to each user u and item v at the time t, which can model the behavior and attributes that change over time. While a new interaction joins the graph, user and item embeddings are updated by DGHP-LISA.

Hawkes process

The Hawkes process is a linear self-excited point process, which assumes that historical events will have the excitation effects on current events. Its behavior is typically modeled by a conditional intensity function $\lambda(t)$, the rate of event occurring at time t given the past events.

The conditional intensity function of Hawkes process can be expressed as follows:

$$\lambda(t) = \mu(t) + \eta \int_{-\infty}^{t} \kappa(t - s) dN(s) \tag{1}$$

where $\mu(t)$ is a constant greater than zero, which represents the base intensity at time t and does not change with the change of historical information. $\kappa(\cdot)$ is the kernel function of Hawkes process. Generally speaking, kernel functions have a variety of forms, including exponential kernel function, Gaussian kernel function and so on. Different kernel functions will affect the decay rate of the model to historical events, thus affecting the prediction ability of the model. For example, in the exponential kernel function, with the increase of time, the influence of historical events on current events gradually decreases. In the Gaussian kernel function, the decay rate becomes smoother with the increase of time. $N(t)$ indicates the number of events that occurred before time t. η is a constant greater than zero, which is used to control the influence of the basic intensity and historical events. Since the Hawkes process is able to model the excitation effects between events to capture

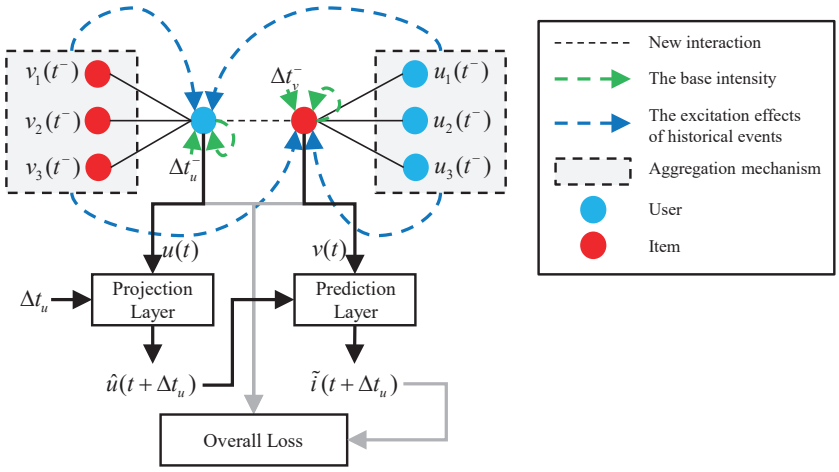

Figure 1 **Overall architecture of the DGHP-LISA.**
Full-size DOI: 10.7717/peerjcs.1368/fig-1

the influence of historical events on current events, it is well suited for modeling user and item representation in dynamic graph.

PROPOSED APPROACH

Figure 1 shows the overall architecture of the proposed model. The interactions between users and items form a dynamic graph over time. When a new user-item interaction event is observed, DGHP-LISA first learns the current embedding of the user and the item on the basis of the dynamic graph Hawkes process simultaneously. Secondly, inspired by the Kalman filter (*Julier & Uhlmann, 1997*; *Hou & Bu, 2021*), we capture the previously updated status and elapsed time through the projection and prediction layer to predict the future embedding of users and items. Thirdly, we calculate the $L2$ distance between the predicted item embedding and all other item embeddings, and then recommend items with the smallest distance to the predicted item embedding. Finally, we use the loss function to jointly optimize the dynamic embedding of users and items to make a more accurate prediction for the next item recommendation.

Embedding update mechanism based on Hawkes process

Hawkes process on dynamic graph

The Hawkes process on the dynamic graph can model the dynamic embedding of users and items. Specifically, whether the user u or the item v participates in the formation of the interaction (u, v) at the time t can be quantified by the conditional intensity of this event.

$$\lambda_u(t) = \mu_u(t) + \alpha \sum_{(u,v',t') \in H_u(t)} \gamma_{hv'}(t')\kappa(t-t') + \beta \sum_{(u',v,t') \in C_u(t)} \gamma_{cu'}(t')\kappa(t-t') \tag{2}$$

$$\lambda_v(t) = \mu_v(t) + \alpha \sum_{(u',v,t') \in H_v(t)} \gamma_{hu'}(t')\kappa(t-t') + \beta \sum_{(u,v',t') \in C_v(t)} \gamma_{cv'}(t')\kappa(t-t') \tag{3}$$

where $\mu_u(t)$ and $\mu_v(t)$ are the basic intensity of the interaction events formed by nodes u or v at the time t, respectively, which are not influenced by historical events on u or v. The basic intensity determines the basic level of the current event, and it can also be understood as the expected probability of the event at the time t. The reasonable setting of the basic intensity can improve the prediction ability and explanation of the model. α and β are positive constants that control the influence of the basic intensity and historical events. If these two parameters are larger, then the impact of historical events on subsequent events will be more lasting. In practical application, we can balance the importance of historical events and the latest events by adjusting α and β.

$H_u(t) = \{(u, v', t') \in I : t' \leq t\}$ and $H_v(t) = \{(u', v, t') \in I : t' \leq t\}$ are the sets of historical interaction events of nodes u or v with respect to time t, respectively. Where v' is the historical interaction neighbor of u, and u' is the historical interaction neighbor of v. $\gamma_{hv'}(t')$ and $\gamma_{hu'}(t')$ respectively represent the influence degree of the historical interaction neighbor v' or u' on the current event at time t'.$C_u(t) = \{(u', v, t') \in I : t' < t\}$ and $C_v(t) = \{(u, v', t') \in I : t' < t\}$ are the sets of historical collaborative interaction events of node u or v with respect to time t, respectively. Where u' is the historical cooperative interaction neighbor of u, v' is the historical cooperative interaction neighbor of v. $\gamma_{cu'}(t')$ and $\gamma_{cv'}(t')$ respectively represent the influence degree of the historical cooperative interaction neighbor u' or v' on the current event at time t'. In the Hawkes process, the time decay effect of historical events on current events is modeled by kernel functions. The kernel function describes the decay process of the excitation effect of historical events with time. Therefore, the influence of historical events on the interaction probability of current events can be explained as the weighted sum of the excitation effect of historical events in time, in which the weight of time is determined by the kernel function. In the DGHP-LISA model, we use the kernel function $\kappa(\cdot)$ to model the time decay effect of historical events on current events, which can control the decay rate of historical events with time.

Next, combined with dynamic graph representation, the conditional intensity in Eqs. (2) and (3) is materialized. LISA$(\cdot)$ is a self-attention module with linear complexity, which can effectively model the importance between update mechanisms by dynamically allocating weights to different embedding update mechanisms. The conditional intensity can be generated from a transfer function f (*Mei & Eisner, 2017*; *Panzarasa, Opsahl & Carley, 2009*), *i.e.,*

$$\lambda_u(t) = f(\text{LISA}(u(t^-), \Delta t_u^-, u_h(t^-), u_c(t^-))) \tag{4}$$

$$\lambda_v(t) = f(\text{LISA}(v(t^-), \Delta t_v^-, v_h(t^-), v_c(t^-))) \tag{5}$$

where $u(t^-)$ and $v(t^-)$ are the latest embeddings of u or v before time t, respectively. Δt_u^- and Δt_v^- respectively represent the time interval between current time t and previous interaction time t^- of the node u or v. For dynamic graph, the node to be updated firstly inherits the influence of the previous state and elapsed time, which is used as the basic intensity to learn the embedding of users and items.

$u_h(t^-)$ and $v_h(t^-)$ represent the aggregate embedding of the historical interaction neighbors of u or v before the time t, respectively. The proposed model establishes a dynamic bipartite graph to simulate the interaction between user and item nodes, which means that the user's historical interaction neighbors are the items that he or she has interacted with, and vice versa. In dynamic recommendation scenarios, the item that a user interacts with reflects the user's recent interest to some extent. Correspondingly, users who are interested in a specific item can be regarded as a part of the item's properties. Therefore, it is necessary to use the historical interaction neighbors of the user or item to learn its embedding.

$u_c(t^-)$ and $v_c(t^-)$ represent the aggregate embedding of the historical collaborative interaction neighbors of u or v before the time t, respectively. In the model, the historical collaborative interaction neighbors of the user or item node are the historical interaction neighbors of the other node involved in the interaction, which capture the collaborative relationship between users and items. Specifically, for a specific item, it may have been purchased by multiple users before the current interaction, and now a new user has purchased the item. It can be assumed that this new user has a collaborative relationship with the users who previously purchased the specific item. Therefore, the update of node embedding in dynamic graph takes into account not only the self-information and historical interaction of nodes, but also the structural information between nodes.

To sum up, the historical interaction aggregation embedding and historical collaborative interaction aggregation embedding of users or items before time t reflect the impact of historical events on current events, which is the key to simulate the excitation effect caused by historical events, combined with the basic intensity to learn the embedding of users and items.

Neighbor aggregation mechanism

It can be seen from Eqs. (4) and (5) that using the Hawkes process on the dynamic graph to model the dynamic embedding of users and items needs to consider the aggregate information of their historical interaction neighbors and historical collaborative interaction neighbors. To aggregate such information, let:

$$u_h(t^-) = \partial(v(t^-), H'_u(t)) \tag{6}$$

$$u_c(t^-) = \partial(u(t^-), C'_u(t)) \tag{7}$$

$$v_h(t^-) = \partial(u(t^-), H'_v(t)) \tag{8}$$

$$v_c(t^-) = \partial(v(t^-), C'_v(t)) \tag{9}$$

where $\partial(\cdot)$ is the aggregator function. $H'_u(t)$ and $H'_v(t)$ represent the historical interaction neighbor embedding set of nodes u or v with respect to time t, respectively. $C'_u(t)$ and

$C'_v(t)$ represent the historical collaborative interaction neighbor embedding set of nodes u or v with respect to time t, respectively.

$$H'_u(t) = \{v''_h(t') | v''_h(t') = \text{LISA}(v'(t') + h(t')), (u, v', t') \in H_u(t)\} \tag{10}$$

$$H'_v(t) = \{u''_h(t') | u''_h(t') = \text{LISA}(u'(t') + h(t')), (u', v, t') \in H_v(t)\} \tag{11}$$

$$C'_u(t) = \{u''_c(t') | u''_c(t') = \text{LISA}(u'(t') + h(t')), (u', v, t') \in C_u(t)\} \tag{12}$$

$$C'_v(t) = \{v''_c(t') | v''_c(t') = \text{LISA}(v'(t') + h(t')), (u, v', t') \in C_v(t)\} \tag{13}$$

where $h(t')$ represents the time embedding of time t'.$v''_h(t')$ and $u''_h(t')$ represent the historical interaction neighbor embedding of nodes u or v at time t' after time encoding, respectively. $u''_c(t')$ and $v''_c(t')$ represent the historical collaborative interaction neighbor embedding of nodes u or v at time t' after time encoding, respectively. As shown in Fig. 2, in order to enable the model to make full use of time information and capture the time decay effects of different historical events on current events, we add time embedding to all neighbor embedding and model the time correlation of different historical events through the LISA module.

For aggregator function $\partial(\cdot)$, we provide the following two candidate aggregator functions for use in neighbor aggregation:

1. Mean aggregator: It is a simple operator to calculate the average value of central node embedding and neighbor node embeddings, which can be regarded as a variant of GCN method (*Hamilton, Ying & Leskovec, 2017*; *Kipf & Welling, 2016*).

$$u_h(t^-) = \frac{1}{1 + |H'_u(t)|}(v(t^-) + \sum_{v''_h(t') \in H'_u(t)} v''_h(t')) \tag{14}$$

$$u_c(t^-) = \frac{1}{1 + |C'_u(t)|}(u(t^-) + \sum_{u''_c(t') \in C'_u(t)} u''_c(t')) \tag{15}$$

$$v_h(t^-) = \frac{1}{1 + |H'_v(t)|}(u(t^-) + \sum_{u''_h(t') \in H'_v(t)} u''_h(t')) \tag{16}$$

$$v_c(t^-) = \frac{1}{1 + |C'_v(t)|}(v(t^-) + \sum_{v''_c(t') \in C'_v(t)} v''_c(t')) \tag{17}$$

2. Attention aggregator: Inspired by the GAT (*Veličković et al., 2017*) model, it uses the attention mechanism to weighted summation of neighbor nodes.

$$u_h(t^-) = \sum_{v''_h(t') \in H'_u(t)} \alpha_h(t') v''_h(t') \tag{18}$$

$$\alpha_h(t') = \frac{\exp(\text{LeakyRelu}(W_\alpha[v(t^-) || v''_h(t')]))}{\sum_{v''_h(t') \in H'_u(t)} \exp(\text{LeakyRelu}(W_\alpha[v(t^-) || v''_h(t')]))} \tag{19}$$

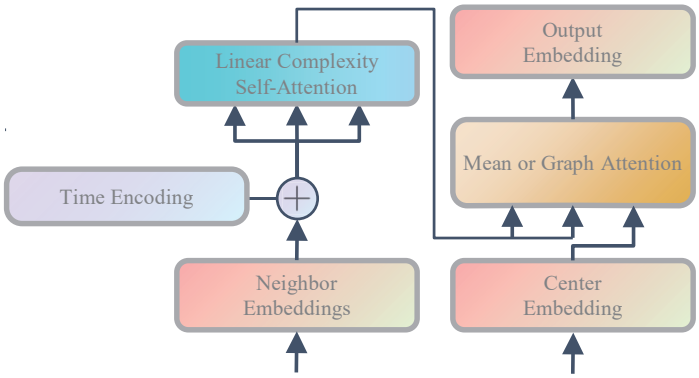

Figure 2 **Neighbor aggregation mechanism.**
Full-size DOI: 10.7717/peerjcs.1368/fig-2

$$u_c(t^-) = \sum_{u''_c(t') \in C'_u(t)} \alpha_c(t') u''_c(t') \tag{20}$$

$$\alpha_c(t') = \frac{\exp(\text{LeakyRelu}(W_\alpha [u(t^-) || u''_c(t')]))}{\sum_{u''_c(t') \in C'_u(t)} \exp(\text{LeakyRelu}(W_\alpha [u(t^-) || u''_c(t')]))} \tag{21}$$

$$v_h(t^-) = \sum_{u''_h(t') \in H'_v(t)} \alpha'_h(t') u''_h(t') \tag{22}$$

$$\alpha'_h(t') = \frac{\exp(\text{LeakyRelu}(W_\alpha [u(t^-) || u''_h(t')]))}{\sum_{u''_h(t') \in H'_v(t)} \exp(\text{LeakyRelu}(W_\alpha [u(t^-) || u''_h(t')]))} \tag{23}$$

$$v_c(t^-) = \sum_{v''_c(t') \in C'_v(t)} \alpha'_c(t') v''_c(t') \tag{24}$$

$$\alpha'_c(t') = \frac{\exp(\text{LeakyRelu}(W_\alpha [v(t^-) || v''_c(t')]))}{\sum_{v''_c(t') \in C'_v(t)} \exp(\text{LeakyRelu}(W_\alpha [v(t^-) || v''_c(t')]))} \tag{25}$$

where $||$ is the concatenation operation and $W_\alpha \in \mathbb{R}^{2d}$ is a weight matrix.

In practical dynamic recommendation scenarios, due to the huge amount of data, neighbor aggregation needs to cost a high computational cost. Therefore, we select a fixed number of neighbors for aggregation and call the number of neighbor nodes selected as aggregator size.

Self-attention mechanism with linear complexity

Let $E_i \in \mathbb{R}^{m \times d_{in}}$ and $E_o \in \mathbb{R}^{m \times d_{out}}$ represent the input and output embedding of the LISA module respectively. m represents the number of embedding. d_{in} and d_{out} represent the dimensions of input and output embedding respectively. Let $Q = E_i W_q$, $K = E_i W_k$ and $V = E_i W_v$ denote the Query, Key and Value matrices generated by linear transformation on input embedding, respectively. Where $W_q \in \mathbb{R}^{d_{in} \times d_k}$, $W_k \in \mathbb{R}^{d_{in} \times d_k}$ and $W_v \in \mathbb{R}^{d_{in} \times d_{out}}$ are all weight matrices. The LISA module generates output embedding using the following

self-attention operations based on linear complexity:

$$E_o = Q(\rho(K^\top)V) \tag{26}$$

where $K^\top$ denotes the matrix transpose of K, and $\rho(\cdot)$ denotes the operation of applying softmax normalization for each row separately. The self-attention operation can be interpreted as first aggregating the features in the V matrix into d_k context vectors using the weights in the $\rho(K^\top)$, and then reassigning the context vectors back to m using the weights in the Q matrix. The computational and memory complexities of this operation are $O(m)$.

This self-attention operation is similar to the attention operation used by *Chen et al. (2018)* or *Shen et al. (2021)*, but we do not use softmax standardization for Q matrix. Normalized Q matrix constrains the output embedding to a convex combination of context vectors, which may limit the expression ability of the self-attention mechanism. Therefore, we remove the softmax normalization of Q matrix, which allows the output embeddings to span the entire subspace of the d_k global context vectors.

Transfer function

The key to materialize the conditional intensity is to fit a transfer function f to the Hawkes process on the dynamic graph. In the past, softplus function and its variants are generally used as transfer function in the study of Hawkes process. To ensure that the output of the transfer function is positive, we instantiate f on the basis of Eqs. (7) and (8) as:

$$\lambda_u(t) = \mathrm{ReLU}(\mathrm{SUM}(\mathrm{LISA}(u(t^-), \Delta t_u^-, u_h(t^-), u_c(t^-)))) \tag{27}$$

$$\lambda_v(t) = \mathrm{ReLU}(\mathrm{SUM}(\mathrm{LISA}(v(t^-), \Delta t_v^-, v_h(t^-), v_c(t^-)))) \tag{28}$$

where the activation function uses ReLU. $\mathrm{SUM}(\cdot)$ means that the features generated by different embedding update mechanisms are fused in a simple addition way.

The connection between transfer function and conditional intensity

A well chosen transfer function f , taking the dynamic graph representations of the node as input, is equivalent to the conditional intensity of the Hawkes process in Eqs. (2) and (3). In this section we formally show the connection.

First, we define the base intensity as a function of the self-information:

$$\mu_u(t) = f_u(u(t^-), \Delta t_u^-) \tag{29}$$

$$\mu_v(t) = f_v(v(t^-), \Delta t_v^-). \tag{30}$$

Next, we define the influence of different historical events on current events as a function of historical interaction neighbors and historical collaborative interaction neighbors.

$$\sum_{(u,v',t') \in H_u(t)} \gamma_{hv'}(t')\kappa(t-t') = f_\gamma(u_h(t^-)) \tag{31}$$

$$\sum_{(u',v,t')\in C_u(t)} \gamma_{cu'}(t')\kappa(t-t') = f_\gamma(u_c(t^-)) \tag{32}$$

$$\sum_{(u',v,t')\in H_v(t)} \gamma_{hu'}(t')\kappa(t-t') = f_\tau(v_h(t^-)) \tag{33}$$

$$\sum_{(u,v',t')\in C_v(t)} \gamma_{cv'}(t')\kappa(t-t') = f_\tau(v_c(t^-)). \tag{34}$$

Given these building blocks, we rewrite the conditional intensity in Eqs. (2) and (3) as:

$$\lambda_u(t) = f_\lambda(u(t^-), \Delta t_u^-, u_h(t^-), u_c(t^-)) \tag{35}$$

$$\lambda_v(t) = f_\pi(v(t^-), \Delta t_v^-, v_h(t^-), v_c(t^-)) \tag{36}$$

where f_λ is a composite function of f_u, f_γ and the summation, and f_π is a composite function of f_v, f_τ and the summation. By choosing the right transfer function f , we further rewrite f_λ and f_π as the composition of f and the LISA module.

Finally, the conditional intensity can be expressed as follows:

$$\lambda_u(t) = (f \circ \mathrm{LISA})(u(t^-), \Delta t_u^-, u_h(t^-), u_c(t^-)) = f(\mathrm{LISA}(u(t^-), \Delta t_u^-, u_h(t^-), u_c(t^-))) \tag{37}$$

$$\lambda_v(t) = (f \circ \mathrm{LISA})(v(t^-), \Delta t_v^-, v_h(t^-), v_c(t^-)) = f(\mathrm{LISA}(v(t^-), \Delta t_v^-, v_h(t^-), v_c(t^-))). \tag{38}$$

Next item prediction

Project and predict the next item embedding

As shown in Fig. 1, in dynamic recommendation, user u interacts with item v at time t, and then interacts with item i at time $t + \Delta t_u$. Our task is to predict items that the user are most likely to interact with before time $t + \Delta t_u$, which is an analogy to link prediction problem in dynamic graph. Specifically, after updating the user and item embedding of time t by using the Hawkes process on the dynamic graph, the DGHP-LISA calculates the projected embedding $\hat{u}(t + \Delta t_u)$ of user u and the predicted embedding $\tilde{i}(t + \Delta t_u)$ of item i through the projection layer and prediction layer in turn. Then, we calculate the L 2 distance between the predicted item embedding and all other item embeddings, and then recommend items with the smallest distance to the predicted item embedding.

1. Projection layer: According to the method suggested in LatentCross (*Beutel et al., 2018*), based on the embedding $u(t)$ of user u at time t and the elapsed time Δt_u, we incorporate time into the projected embedding *via* Hadamard product. The formula is as follows:

   $$\hat{u}(t + \Delta t_u) = u(t) \odot (1 + W_{pro}\Delta t_u) \tag{39}$$

where W_{pro} is used to convert Δt_u into a time-context vector. We initialize W_{pro} by a 0-mean Gaussian. The vector $1 + W_{pro}\Delta t_u$ is used to scale the past user embedding. When $\Delta t_u = 0$, the projected embedding is the same as the input embedding vector. The larger the value of Δt_u, the more the projected embedding differs from the input embedding and the projected embedding drifts over time.

2. Prediction layer: After obtaining the projected embedding $\hat{u}(t + \Delta t_u)$ of user u, we use the prediction layer function to learn the future embedding of item i represented as $\tilde{i}(t + \Delta t_u)$. We make this prediction based on the current user u and its interaction item v immediately before time $t + \Delta t_u$. The formula is as follows:
$$\tilde{i}(t + \Delta t_u) = W_{pre}[\hat{u}(t + \Delta t_u)||\bar{u}||v(t)||\bar{v}]. \tag{40}$$
As shown in Eq. (40), we use both the static and dynamic embeddings to predict the static and dynamic embedding of item i at time $t + \Delta t_u$, and W_{pre} represents the weight matrix.

Loss function

DGHP-LISA is trained to minimize the $L2$ distance between the predicted item embedding and the ground truth item's embedding at every interaction. The loss function is as follows:

$$L = \sum_{(u,v,t)\in I} \left\|\tilde{v}(t) - [v(t^-)\|\bar{v}]\right\|_2 + \lambda_u \left\|u(t) - u(t^-)\right\|_2 + \lambda_v \left\|v(t) - v(t^-)\right\|_2. \tag{41}$$

The first loss term minimizes the predicted embedding error. Since the items' and users' properties tend to be stable in a short time, the last two terms are added to regularize the loss and prevent the consecutive dynamic embeddings of a user and item to vary too much, respectively. I is a sequence of user-item interaction events in chronological order. λ_u and λ_v are scaling parameters, which are used to ensure the losses are in the same range.

Optimization and training

In model training, we used a gradient-based Adam optimizer to optimize the parameters and use the same method of constructing batches as in *Kumar, Zhang & Leskovec (2019)* to speed up the training process. All interactions in each batch created by this method do not share any common nodes. DGHP-LISA works iteratively between the selection and removal steps. In the selection step, the method creates a new batch by selecting the largest set of edges at the earliest time. In the removal step, the previously selected edges are removed from the dynamic graph. Thus, each batch built by this method is parallelizable and maintains the sequential dependencies of all interactions.

Figure 3 shows a dynamic graph of four users interacting with three items over time, where each dotted line represents the interaction associated with time.

As can be seen from Fig. 3, a total of three batches are constructed using this batch construction method, which is 50% less than when a single interaction is assigned to a batch. It is worth noting that each batch satisfies:

1. Each user and item appears at most once in every batch;
2. For the same user or item, earlier interactions are assigned to earlier batches.

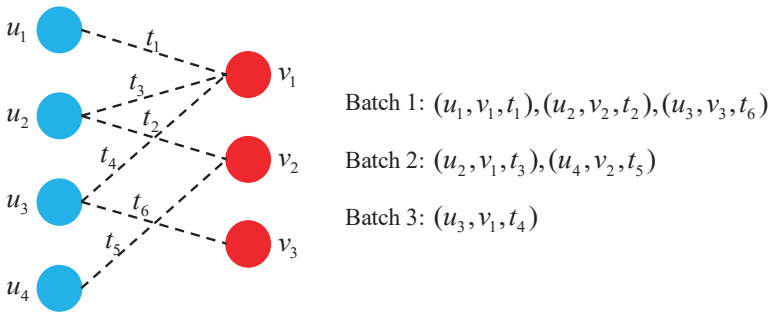

Figure 3 **Batch construction in dynamic graph.**
Full-size DOI: 10.7717/peerjcs.1368/fig-3

Table 1 **The details of datasets.**

Data	Users	Items	Interactions	Action repetition
Yelp	9,081	1,000	77,546	0.5%
LastFM	1,000	1,000	1293,103	8.6%
GTD	566	744	99,043	22.7%

EXPERIMENTS

Dataset and preprocessing

In order to evaluate the performance of DGHP-LISA, we conduct experiments on three public datasets. The details of the datasets are shown in Table 1.

1. Yelp: This dataset is a subset of Yelp's businesses, reviews, and user data for use in personal, educational, and academic purposes. We first selected top 1,000 businesses with most number of reviews and users who made at least five reviews on the selected businesses. This resulted in 9,081 users and 77,546 interactions. In Yelp, a user interacts with the same business consecutively only in 0.5% interactions. In the experiment, this dataset is divided into training set, verification set and test set according to the proportion of 80%, 10% and 10%.
2. LastFM: This dataset is a widely used dataset that contains one month's song listening information (*Hidasi & Tikk, 2012*). We selected all 1,000 users and the 1,000 most listened songs, resulting in a total of 1,293,103 interactions. Note that users only listened to the same song continuously in 8.6% of the interactions. The training set, validation set, and test set are divided in the same proportion as the Yelp dataset.
3. GTD: The global terrorism database is considered to be the most comprehensive database covering terrorist incidents in the field of global terrorism so far, including more than 200,000 terrorist incidents committed by about 3,000 terrorist organizations around the world since 1970. In order to provide early warning of terrorist attacks and reduce risks, we use DGHP-LISA to predict the next province or state where terrorist organizations may attack at a specific time. In the experiment, we deleted all events with uncertain dates, as well as events with unknown organization and location

of the terrorist attacks. At the same time, we remove all the terrorist organizations with fewer than ten occurrences and all the locations where were attacked with less than ten occurrences. Finally, a total of 99,043 events were screened, of which a terrorist organization interacts with the same location of attack consecutively in 38% interactions. In the experiment, the data is split by time, and we train all models on the first 70% interactions, validate on the next 10%, and test on the last remaining interactions.

Experimental setup

Evaluation metrics

We use the following two evaluation metrics for experiments:

1. Mean Reciprocal Rank (*MRR*) is the average of the reciprocal rank of the first positive example in all user recommendation lists. This indicator can measure the performance of the model with respect to the ranking list of items. Higher *MRR* score means the ground truth item tends to have higher rank positions in the predicted item lists. In some scenarios, such as ranking search results for specific users, the improvement of *MRR* can significantly improve the user experience, especially for items with high importance in ranking results. To calculate *MRR*, we use equation:
$$MRR=\frac{1}{|I|}\sum_{i\in I}\frac{1}{rank_i} \tag{42}$$
where $i \in I$ represents traversing all interactions and $rank_i$ represents the location of the ground truth item in the recommendation list for the i-th interaction.
2. *Recall@k* is the fraction of interactions in which the ground truth item is ranked in the top k. This metrics pays more attention to the integrity and diversity of recommendation results, and it is more suitable to be used in the recommendation system. Because the recommendation system needs to ensure that it covers as many items as possible in the user's area of interest, so as to improve user satisfaction and participation. In some scenarios, such as e-commerce and news recommendation, the improvement of *Recall@k* can significantly improve the actual effect of the recommendation system, increase sales or user stickiness and so on. Formally, *Recall@k* is defined as:
$$Recall@k=\frac{n_{hit}}{|I|} \tag{43}$$
where n_{hit} is the number of ground truth items that are among the top-k recommendation list, and $|I|$ is the number of all test cases.

Parameter setting

We use Pytorch to implement DGHP-LISA, and the hyper-parameters are determined by the performance of the model on the verification set. For all algorithms, we use 128-dimensional dynamic embeddings and randomly initialize user and item embeddings with a Gaussian distribution with mean 0 and variance 1. Adam optimizer with learning rate 1e−3, $L2$ penalty 1e−5 is adopted in our model. Static embeddings all use one-hot vectors. The scaling parameters λ_u and λ_v in loss function are set to 1. All algorithms run for 50 rounds, and the corresponding test set is selected according to the best verification set. All experiments are run independently in the same experimental environment with Intel(R)

Xeon(R) Gold 5118 host and NVIDIA Tesla V100-SXM2-32GB GPU. For comparison methods, we mostly use the default hyperparameters of the original paper.

Baselines

To evaluate DGHP-LISA, we compare it with the following six baselines:

- LSTM (*Hochreiter & Schmidhuber, 1997*): It is a special RNN that captures long-term dependencies.
- Time-LSTM (*Zhu et al., 2017*): It adds two time gates to the standard LSTM to model the impact of time intervals on users' current and long-term behavior.
- RRN (*Wu et al., 2017*): Based on the idea of matrix decomposition, it uses RNN to learn the dynamic embedding of users and items.
- DeepCoevolve (*Dai et al., 2016*): It is based on co-evolutionary point process algorithms. According to (*Kumar, Zhang & Leskovec, 2019*), 10 negative samples are used in each interaction.
- JODIE (*Kumar, Zhang & Leskovec, 2019*): It is a coupled RNN model that is used to learn the dynamic feature representation of users and items.
- DGCF (*Li et al., 2020*): It is a state-of-the-art model in dynamic recommendation problem, which is a new framework that utilizes dynamic graph to capture the collaborative and historical sequence relationships between users and items.

Next item prediction experiment

For next item prediction, Table 2 compares the results of DGHP-LISA with six baselines on three datasets. In the experiment, DGHP-LISA uses a attention aggregator with an aggregator size of 20. The bold and underlined numbers mean the best and second-best results on each dataset and metric, respectively. "Improvement" denotes the performance improvement of DGHP-LISA over the best baseline.

From the experimental results, it can be observed:

1. DGHP-LISA outperforms all baselines on both metrics of the three datasets. Especially on the Yelp dataset, compared with the best baseline, the improvement of DGHP-LISA on *MRR* is 14.3% and the improvement on *Recall@10* is 23.6%. Also, compared with state-of-the-art baseline DGCF, the improvements on GTD, LastFM, and Yelp are in an increasing order, which are consistent with the repetitive action pattern in the datasets. The main reason for the improvement may be that DGHP-LISA explicitly captures the historical interaction information and historical collaborative interaction information. For example, users in Yelp datasets tend to comment on different businesses with similar categories, which results in a low action repetition. In this case, our model can consider both the historical interaction sequence of users and the similar interaction sequence of other users. The results show that DGHP-LISA can well deal with low action repetition situation.
2. DGHP-LISA performs significantly better than DeepCoevolve on three datasets. It shows that DGHP-LISA has a more appropriate conditional intensity function, which can better model the impact of historical events on the embedding update process of users or items.

Table 2 Performance comparison of different methods on three datasets.

Models	Yelp		LastFM		GTD	
	MRR	*Recall@* 10	*MRR*	*Recall@* 10	*MRR*	*Recall@* 10
LSTM	0.006	0.011	0.062	0.119	0.124	0.211
Time-LSTM	0.008	0.019	0.068	0.137	0.267	0.452
RRN	0.013	0.023	0.089	0.182	0.402	0.643
DeepCoevolve	0.004	0.009	0.019	0.039	0.051	0.085
JODIE	0.046	0.082	0.195	0.307	0.496	0.764
DGCF	0.077	0.144	0.321	0.456	0.509	0.785
DGHP-LISA	**0.088**	**0.178**	**0.328**	**0.474**	**0.510**	**0.803**
Improvement	14.3%	23.6%	2.2%	3.9%	0.2%	2.3%

Notes.
The bold and underlined values indicate the best and second-best results on each dataset and metric, respectively.

3. DGHP-LISA and DGCF are superior to other baselines on three datasets, which indicates that the modeling of topological structure information is critical for learning dynamic graphs.
4. Considering the low density of user-item matrix in the three datasets, DGHP-LISA has certain advantages in dealing with the problem of data sparsity in the field of dynamic recommendation. First of all, the method makes use of the dynamic interactive information between the user and the item in the time series bipartite graph to make recommendations. This dynamic interactive information can provide more feature information and increase the accuracy of model prediction, so as to make up for the lack of data sparsity. Secondly, the DGHP-LISA models the changes of users' interests and the evolution of item attributes at the same time by introducing Hawkes process, so as to better depict the co-evolution between users and items. It can be proved by experiments that this method can further enhance the representation of learning nodes in order to make full use of the limited data and make better recommendations.

Experimental analysis and discussion

Ablation study

To verify the effectiveness of each module in the model, we implemented several variants of DGHP-LISA and conduct the next item prediction task on three datasets. In the experiment, DGHP-LISA uses a attention aggregator with an aggregator size of 20. First of all, we use only the basic intensity in the node embedding update framework based on the dynamic graph Hawkes process, and then gradually add the excitation effect of historical events, the time embedding of neighbor nodes, and the LISA module to form DGHP-LISA. The specific process is described as follows:

- variant *a* (basic intensity): We only use basic intensity to model in the node embedding update framework based on dynamic graph Hawkes process.
- variant *b* (+excitation effect of historical events): We add the modeling of the influence of different historical interaction events and historical collaborative interaction events on the current events.

- variant *c* (+time embedding of neighbor nodes): We add time embedding to all historical interaction neighbors and historical collaborative interaction neighbors.
- variant *d* (+ LISA): LISA module is added to the time encoding of neighbor nodes and the feature fusion of node embeddings.

Table 3 presents that the key modules in DGHP-LISA are effective for the next item prediction task. Specifically, modeling the influence of historical events on current events in the node embedding update framework based on dynamic graph Hawkes process is helpful to capture the excitation effects of different historical events. We add time embedding to all historical interaction neighbors and historical collaborative interaction neighbors, which is helpful to capture the time decay effect of different historical events on current events. We add the LISA module to the time encoding of neighbor nodes, which effectively captures the time correlation of different historical events. At the same time, we add the LISA module to the feature fusion of node embeddings , which effectively models the importance between update mechanisms. In conclusion, the ablation experiment presents the effectiveness of each module in our model.

Impact of different aggregator functions

In order to study the impact of different aggregator functions on the performance of the model, we test the effectiveness of mean and attention aggregator functions respectively. In the experiment, the size of the aggregator is 20. It can be seen from Table 4 that the attention aggregator is better than the mean aggregator on three datasets. It is proved that the attention aggregator can update the user or item embedding by selecting the information which is more beneficial to the prediction task in multiple historical interaction neighbors or historical collaborative interaction neighbors. However, the mean aggregator has the same weight when aggregating neighbor information, which may ignore the most influential nodes.

Impact of different aggregator sizes

In order to verify the impact of different aggregator sizes on the model performance, we use attention aggregator to set different aggregator sizes to evaluate the model. In the experiment, DGHP-LISA uses a attention aggregator, and the aggregator size is set to 20, 40, 60 and 80, respectively. As shown in Table 5, as the size of the aggregator increases, the performance of the model on three datasets decreases at first and then increases. When the size of aggregator is 20, the performance of the algorithm is the best. This shows that the increase of aggregator size may bring a lot of neighbor information redundancy, which is not helpful to modeling. Therefore, the accuracy of the model and the speed of training can be improved by reducing the size of the aggregator.

CONCLUSIONS

We propose a new dynamic graph Hawkes process based on linear complexity self-attention and successfully apply it to the next item prediction problem. This model proposes an effective method to model the dynamic embedding of users and items in dynamic graph. Specifically, on the node embedding update framework based on Hawkes process, we

Table 3 Comparison of different variant models.

Variants	Yelp		LastFM		GTD	
	MRR	*Recall@* 10	*MRR*	*Recall@* 10	*MRR*	*Recall@* 10
a	0.077	0.149	0.322	0.460	0.508	0.795
b	0.079	0.161	0.324	0.461	0.508	0.796
c	0.081	0.163	0.325	0.462	0.509	0.798
d	0.088	0.178	0.328	0.474	0.510	0.803

Table 4 Impact of different aggregator functions.

Aggregator	Yelp		LastFM		GTD	
	MRR	*Recall@* 10	*MRR*	*Recall@* 10	*MRR*	*Recall@* 10
Mean	0.087	0.176	0.326	0.472	0.509	0.802
Attention	0.088	0.178	0.328	0.474	0.510	0.803

Table 5 Impact of different aggregator sizes.

Aggregator size	Yelp		LastFM		GTD	
	MRR	*Recall@* 10	*MRR*	*Recall@* 10	*MRR*	*Recall@* 10
20	0.088	0.178	0.328	0.474	0.510	0.803
40	0.086	0.176	0.327	0.474	0.508	0.802
60	0.087	0.175	0.326	0.473	0.509	0.801
80	0.088	0.177	0.328	0.474	0.510	0.802

add time embedding to all neighbor nodes to capture the time decay effects of different historical events on current events and model the excitation effects of different historical events. At the same time, we add LISA module to the time encoding of neighbor nodes and the feature fusion of node embeddings respectively, which effectively captures the time correlation of different historical events and the dynamic correlation between different update mechanisms. Finally, extensive experiments are carried out on three real-world datasets to prove that DGHP-LISA achieves the most advanced performance. In the future research work, a non-recursive network structure can be designed to further improve the training speed without affecting the accuracy of the model. Another direction of innovation is to explore the combination of other dynamic graph structure and recommender system except bipartite graph In addition, it is also possible to explore how to consider the impact of negative historical events on current events in DGHP-LISA to more accurately reflect user interest changes and item attribute evolution.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the National Natural Science Foundation of China-China State Railway Group Co., Ltd. Railway Basic Research Joint Fund (Grant No.U2268217)

and the Scientific Funding for China Academy of Railway Sciences Corporation Limited (No.2021YJ183). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
The National Natural Science Foundation of China-China State Railway Group Co., Ltd. Railway Basic Research Joint Fund: No.U2268217.
The Scientific Funding for China Academy of Railway Sciences Corporation Limited: No.2021YJ183.

Competing Interests

Xiaojun Lv is employed by China Academy of Railway Sciences Corporation Limited.

Author Contributions

- Zhiwen Hou conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Xiaojun Lv performed the experiments, authored or reviewed drafts of the article, and approved the final draft.
- Yuchen Zhou analyzed the data, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Lingbin Bu analyzed the data, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Qiming Ma conceived and designed the experiments, prepared figures and/or tables, and approved the final draft.
- Yifan Wang performed the experiments, prepared figures and/or tables, and approved the final draft.
- Fanliang Bu conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The data and code are available in the Supplemental Files.

Publicly available datasets were analyzed in this study. We evaluated our algorithm on three datasets: Yelp, LastFM, and GTD:

- https://www.yelp.com/dataset;
- http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html;
- https://www.start.umd.edu/gtd.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.1368#supplemental-information.

REFERENCES

Baytas IM, Xiao C, Zhang X, Wang F, Jain AK, Zhou J. 2017. Patient subtyping via time-aware LSTM networks. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. New York: ACM, 65–74.

Beutel A, Covington P, Jain S, Xu C, Li J, Gatto V, Chi EH. 2018. Latent cross: making use of context in recurrent recommender systems. In: *Proceedings of the eleventh ACM international conference on web search and data mining*. New York: ACM, 46–54.

Chen Y, Kalantidis Y, Li J, Yan S, Feng J. 2018. A$\hat{2}$-nets: double attention networks. In: *Advances in neural information processing systems*. Redhook: Curran Associate, 350–359.

Covington P, Adams J, Sargin E. 2016. Deep neural networks for youtube recommendations. In: *Proceedings of the 10th ACM conference on recommender systems*. New York: ACM, 191–198.

Dai H, Wang Y, Trivedi R, Song L. 2016. Deep coevolutionary network: embedding user and item features for recommendation. ArXiv preprint. arXiv:1609.03675.

Hamilton W, Ying Z, Leskovec J. 2017. Inductive representation learning on large graphs. ArXiv preprint. arXiv:1706.02216.

Hawkes AG. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* **58(1)**:83–90 DOI 10.1093/biomet/58.1.83.

Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. 2015. Session-based recommendations with recurrent neural networks. ArXiv preprint. arXiv:1511.06939.

Hidasi B, Tikk D. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In: Flach PA, De Bie T, Cristianini N, eds. *Machine learning and knowledge discovery in databases. ECML PKDD 2012. Lecture Notes in Computer Science, vol 7524*, Berlin, Heidelberg: Springer DOI 10.1007/978-3-642-33486-3_5.

Hochreiter S, Schmidhuber J. 1997. Long short-term memory. *Neural Computation* **9(8)**:1735–1780 DOI 10.1162/neco.1997.9.8.1735.

Hou Z, Bu F. 2021. A small UAV tracking algorithm based on AIMM-UKF. *Aircraft Engineering and Aerospace Technology* **93**:579–591 DOI 10.1108/AEAT-01-2019-0013.

Julier SJ, Uhlmann JK. 1997. New extension of the Kalman filter to nonlinear systems. In: *Signal processing, sensor fusion, and target recognition VI, vol. 3068*. Bellingham: SPIE, 182–193.

Kipf TN, Welling M. 2016. Semi-supervised classification with graph convolutional networks. ArXiv preprint. arXiv:1609.02907.

Koren Y. 2009. Collaborative filtering with temporal dynamics. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York: ACM, 447–456.

Kumar S, Zhang X, Leskovec J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. New York: ACM, 1269–1278.

Li J, Ren P, Chen Z, Ren Z, Lian T, Ma J. 2017. Neural attentive session-based recommendation. In: *Proceedings of the 2017 ACM on conference on information and knowledge management*. New York: ACM, 1419–1428.

Li X, Zhang M, Wu S, Liu Z, Wang L, Philip SY. 2020. Dynamic graph collaborative filtering. In: *2020 IEEE international conference on data mining (ICDM)*. Piscataway: IEEE, 322–331.

Liu Q, Wu S, Wang D, Li Z, Wang L. 2016a. Context-aware sequential recommendation. In: *2016 IEEE 16th international conference on data mining (ICDM)*. Piscataway: IEEE, 1053–1058.

Liu Q, Wu S, Wang L. 2017. Multi-behavioral sequential prediction with recurrent log-bilinear model. *IEEE Transactions on Knowledge and Data Engineering* **29(6)**:1254–1267 DOI 10.1109/TKDE.2017.2661760.

Liu Q, Wu S, Wang L, Tan T. 2016b. Predicting the next location: a recurrent model with spatial and temporal contexts. In: *Thirtieth AAAI conference on artificial intelligence*. Palo Alto: AAAI,.

Mei H, Eisner JM. 2017. The neural hawkes process: a neurally self-modulating multivariate point process. In: *NIPS*. 6757–6767.

Panzarasa P, Opsahl T, Carley KM. 2009. Patterns and dynamics of users' behavior and interaction: network analysis of an online community. *Journal of the American Society for Information Science and Technology* **60(5)**:911–932 DOI 10.1002/asi.21015.

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. 2012. BPR: Bayesian personalized ranking from implicit feedback. ArXiv preprint. arXiv:1205.2618.

Rendle S, Freudenthaler C, Schmidt-Thieme L. 2010. Factorizing personalized markov chains for next-basket recommendation. In: *Proceedings of the 19th international conference on world wide web*. 811–820.

Shen Z, Zhang M, Zhao H, Yi S, Li H. 2021. Efficient attention: attention with linear complexities. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. Piscataway: IEEE, 3531–3539.

Tan YK, Xu X, Liu Y. 2016. Improved recurrent neural networks for session-based recommendations. In: *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

Trivedi R, Farajtabar M, Biswal P, Zha H. 2019. Dyrep: learning representations over dynamic graphs. In: *International conference on learning representations*.

Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. 2017. Graph attention networks. ArXiv preprint. arXiv:1710.10903.

Wang X, He X, Wang M, Feng F, Chua T-S. 2019. Neural graph collaborative filtering. In: *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information retrieval*. New York: ACM, 165–174.

Wang Y, Du N, Trivedi R, Song L. 2016. Coevolutionary latent feature processes for continuous-time user-item interactions. In: *Proceedings of the 30th international conference on neural information processing systems. NIPS'16*. Red Hook, NY, USA: Curran Associates Inc., 4554–4562.

Wang Z, Wei W, Cong G, Li X-L, Mao X-L, Qiu M. 2020. Global context enhanced graph neural networks for session-based recommendation. In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. New York: ACM, 169–178.

Wu C-Y, Ahmed A, Beutel A, Smola AJ, Jing H. 2017. Recurrent recommender networks. In: *Proceedings of the tenth ACM international conference on web search and data mining*. New York: ACM, 495–503.

Wu Q, Gao Y, Gao X, Weng P, Chen G. 2019a. Dual sequential prediction models linking sequential recommendation and information dissemination. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. New York: ACM, 447–457.

Wu S, Liu Q, Bai P, Wang L, Tan T. 2016. SAPE: a system for situation-aware public security evaluation. In: *Thirtieth AAAI conference on artificial intelligence*. Palo Alto: AAAI.

Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T. 2019b. Session-based recommendation with graph neural networks. In: *Proceedings of the AAAI conference on artificial intelligence, vol. 33*. Palo Alto: AAAI, 346–353.

Xu C, Zhao P, Liu Y, Sheng VS, Xu J, Zhuang F, Fang J, Zhou X. 2019. Graph contextualized self-attention network for session-based recommendation. In: *IJCAI, vol. 19*. 3940–3946.

You J, Wang Y, Pal A, Eksombatchai P, Rosenburg C, Leskovec J. 2019. Hierarchical temporal convolutional networks for dynamic recommender systems. In: *The world wide web conference*. 2236–2246.

Zhang Y. 2019. ATTAIN: attention-based time-aware LSTM networks for disease progression modeling. In: *Proceedings of the 28th international joint conference on artificial intelligence (IJCAI-2019)*. Macao, China, 4369–4375.

Zhu Y, Li H, Liao Y, Wang B, Guan Z, Liu H, Cai D. 2017. What to do next: modeling user behaviors by time-LSTM. In: *IJCAI, vol. 17*. 3602–3608.