World Wide Web
https://doi.org/10.1007/s11280-023-01205-8

FPGN: follower prediction framework for infectious disease prevention

Jianke Yu[1] · Xianhang Zhang[2] · Hanchen Wang[1] · Xiaoyang Wang[2] · Wenjie Zhang[2] · Ying Zhang[1,3]

Received: 25 June 2023 / Revised: 15 August 2023 / Accepted: 26 August 2023

Abstract
In recent years, how to prevent the widespread transmission of infectious diseases in communities has been a research hot spot. Tracing close contact with infected individuals is one of the most severe problems. In this work, we present a model called Follower Prediction Graph Network (FPGN) to identify high-risk visitors, which is known as follower prediction. The model is designed to identify visitors who may be infected with a disease by tracking their activities at the exact location of infected visitors. FPGN is inspired by the state-of-the-art temporal graph edge prediction algorithm TGN and draws on the shortcomings of existing algorithms. It utilizes graph structure information based on (α, β)-core, time interval statistics by using the statistics of timestamp information, and a GAT-based prediction module to achieve high accuracy in follower prediction. Extensive experiments are conducted on two real datasets, demonstrating the progress of FPGN. The experimental results show that FPGN can achieve the highest results compared with other SOTA baselines. Its AP scores are higher than 0.46, and its AUC scores are higher than 0.62.

Keywords Follower prediction · Temporal bipartite graphs · Graph neural networks

✉ Hanchen Wang
hanchen.wang@uts.edu.au

Jianke Yu
jianke.yu@student.uts.edu.au

Xianhang Zhang
xianhang.zhang@unsw.edu.au

Xiaoyang Wang
xiaoyang.wang1@unsw.edu.au

Wenjie Zhang
wenjie.zhang@unsw.edu.au

Ying Zhang
ying.zhang@uts.edu.au

1 University of Technology Sydney, Sydney, Australia

2 University of New South Wales, Sydney, Australia

3 Zhejiang Gongshang University, Hangzhou, China

Published online: 16 September 2023

1 Introduction

In recent years, the outbreak of infectious diseases has caused great harm to human health and social stability. To prevent and control the spread of infectious diseases, it is necessary to predict and control the high-risk population accurately. In this regard, graph analysis has become an essential tool for epidemiologists [14, 32]. Temporal bipartite graphs represent the relationships between two sets of vertices while tracking their evolution over time. As a result, this structure is particularly useful in analyzing infectious disease transmission patterns. By analyzing these temporal bipartite graphs, we can identify gaps and clusters that may indicate potential transmission.

This paper focuses on identifying high-risk visitors who may be infected with a disease by tracking their activities at the same location as infected visitors. The identification allows public health authorities to take proactive measures to prevent the spread of infectious diseases. The objective is to predict the likelihood that two visitors (i.e., a follower and a leader) will visit the same location within a specific time window. This challenge is called the follower prediction problem.

Example An example of the following behavior is shown in Figure 1. We assume that the time window to identify the following behavior in this figure is half an hour, i.e., if two visitors visit the same venue within half an hour, the second visitor is identified as the follower of the first visitor. We also assume that Bob (u_1) is an infected visitor. We can see that Bob went to the cafe (v_1) at 8 o'clock in the morning, and another visitor Sam (u_2) arrived within ten minutes after Bob's arrival. Therefore, we consider Sam to be one of Bob's followers. Although Lisa (u_3) and Bob also arrived at Gym(v_2) within ten minutes of each other, Lisa arrived at the venue before Bob did. Jack(u_4) arrived at the restaurant(v_4) three hours after Bob did. Therefore, neither Lisa nor Jack is a follower of Bob.

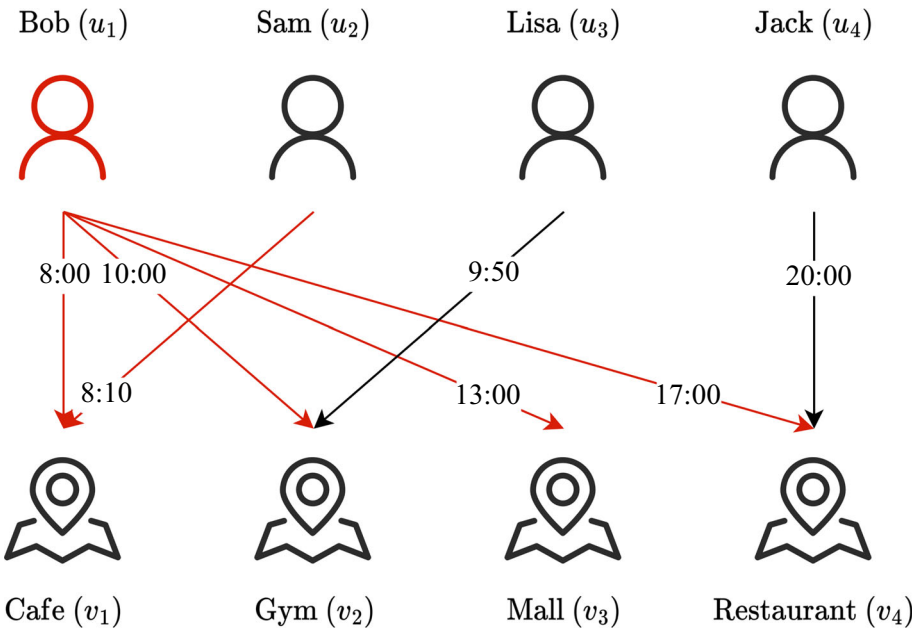

Figure 1 An Example of Following Behavior

Challenge The difficulty in solving this problem lies in two aspects: on the one hand, due to the special nature of the bipartite graph structure, there will be no connections between vertices of the same type in the dataset, while the goal of follower prediction is precisely to predict connections between vertices of the same type. On the other hand, the conditions for constituting the following behavior are strict. They require simultaneous consideration of both cases where two vertices are connected to the same vertex and their time difference for connecting.

To the best of our knowledge, there is no existing method proposed to solve the follower prediction problem. One possible solution is to use a high-performence GNN model [4, 15, 19, 42, 49] trained on datasets for this problem. However, these generic models do not effectively utilize timestamp information and have limited accuracy. A related problem of the follower prediction problem is edge prediction in temporal graphs, which has been addressed by various algorithms [3, 20, 33, 40, 47]. These methods accurately predict edges on temporal graphs but either rely solely on timestamps as sampling criteria or only learn individual activity information for each vertex. Consequently, models designed for other purposes may struggle with this specific follower prediction problem and even perform worse than general GNN models due to being misled by their original goals.

In order to better solve the problem of follower prediction, inspired by the state-of-the-art temporal graph edge prediction algorithm TGN [33] and drawing on the shortcomings of existing algorithms, we propose a model called Follower Prediction Graph Network (FPGN). TGN significantly contributes to preserving temporal information by learning the memory of vertices. Therefore, the vertex memory obtained by TGN can help us solve the problem of predicting connections between same-type vertices on bipartite graphs. However, TGN cannot be directly applied to this problem. When TGN updates the memory of a vertex, it refers to its previous memories, allowing for sampling train sets based on chronological order. In other words, TGN updates vertex memories while sampling based on downstream task requirements. On the other hand, to predict followers accurately, it is necessary to anticipate the connections between same-type vertices in the bipartite graph that have been updated after all timestamps in the training set. This makes it impossible for TGN to be directly used for solving follower problems. Motivated by TGN and learned from his shortcomings, we design our model by optimizing the structure of TGN to let it update memories of all vertices beforehand and then train according to downstream needs.

Besides, FPGN can also consider the correlation information between vertex activities to enhance the model's ability to solve follower prediction problems. Specifically, in addition to learning which vertices will be followers of their leaders, FPGN will further analyze the time window in which these following behaviors occur, allowing the model to grasp the degree of association or closeness between these followers and their leaders. This strategy can help FPGN achieve the ability to predict the following behaviors whose conditions are strict.

Moreover, when some followers often follow several other fixed leaders, the density of the graph composed of them will be relatively high. Therefore, information about the structure of the graph, especially the density, is essential in the follower prediction problem. A typical way to find dense subgraphs on a bipartite graph is biclique [1, 58]. More specifically, biclique aims to find subgraphs on a bipartite graph where any vertex connects to all of the vertices belonging to another class. Thus, it can be assumed that any vertex in the biclique is closely related to all other vertices in this subgraph. However, on the one hand, biclique is an NP-hard problem that is difficult to obtain quickly by conventional ways [26, 27, 31]. On the other hand, biclique is too strict to provide richer structural information for the model. Therefore,

we choose to use the more flexible (α, β)-core [23] to extract the structural information of the graph. The advantage of using (α, β)-core is that it allows the size of α and β to be controlled according to actual needs so that the model can flexibly obtain the required structural information while reducing computational costs. With the help of (α, β)-core and time interval statistics, combined with the mapping method based on UniMP [37], FPGN can effectively solve the problem of difficult-to-predict relationships between vertices of the same type on temporal bipartite graphs, obtain highly accurate follower prediction results.

Contributions The contributions of this paper are summarized as follows:

- As far as we know, FPGN is the first model proposed to solve the problem of follower prediction.
- By relying on improvements to TGN and carefully designed modules that make good use of temporal statistical information and graph structure information, FPGN has achieved extremely high accuracy in follower prediction.
- Extensive experiments are carried out on two real datasets, and the results proved the effectiveness of FPGN.

2 Background

2.1 Preliminaries

The temporal bipartite graph is denoted as $G = (U, V, T, A)$, where $U = \{u_1, u_2, \dots\}$ is a visitor set; $V = \{v_1, v_2, \dots\}$ is a location set; $T = \{t_{(u_1,v_1)}, t_{(u_2,v_1)}, \dots\}$ is a set of times when a visitor arrives at a location, where $t_{(u_1,v_1)} = \{t^{(1)}_{(u_1,v_1)}, t^{(2)}_{(u_1,v_1)}, \dots\}$ is a set of timestamps to record each timestamp of a visitor arriving at the exact location; $A = \{\mathbf{A}^{t_1}, \mathbf{A}^{t_2}, \dots\}$ is the set of adjacency matrix, where $\mathbf{A}^{t_1}$ denotes all visitor records for at time t_1. If u and v connect with each other at time t, then $\mathbf{A}^t_{u,v} = 1$, otherwise $\mathbf{A}^t_{u,v} = 0$.

Definition 1 (Minimum Following Time Interval) Given a leader u, a follower u', and a location v that the follower follows the leader to reach, the definition of the minimum following time interval:

$$\Delta(u, u', v) = min\{t - t' | t \in t_{(u,v)}, t' \in t_{(u',v)}, t' - t \geq 0\} \tag{1}$$

Definition 2 ((α, β)-core) Given a bipartite graph $G = (U, V)$ and two integers α and β, the (α, β)-core of G is made up of two subsets of vertices: $U' \in U$ and $V' \in V$. These subsets induce a bipartite subgraph g that is the largest subgraph of G where all vertices in U' have a degree of at least α and all vertices in V' have a degree of at least β.

2.2 Problem statement

Given a temporal bipartite graph G, the task of follower prediction is to predict whether u' will go to the location v where u has visited in a certain time window $\tau_t > 0$. That is, for visitor u and u', if $\{0 \leq \Delta(u, u', v) \leq \tau_t | v \in V\} \neq \phi$, then u' is a follower of leader u.

Generally speaking, graph neural networks can be summarized by the following formula:

$$\mathbf{h}_u^{(k)} = \mathcal{COM}^{(k)}(\mathbf{h}_u^{(k-1)}, \mathcal{AGG}^{(k)}\{\mathbf{h}_{u'}^{(k)}; u' \in N(u)\}), \tag{2}$$

where $\mathbf{h}_u^{(k)}$ represents the hidden representation of vertex u at layer k, this vector captures important information about the vertex and its relationships with other vertices in the graph; $\mathbf{h}_u^{(k-1)}$ refers to the hidden representation of vertex u at the previous layer $(k-1)$. In other words, it is the input to the current layer k; $\mathcal{AGG}^{(k)}\{\mathbf{h}_{u'}^{(k)}; u' \in N(u)\}$ is the aggregation function, which combines the hidden representations of neighboring vertices $u' \in N(u)$ to produce a summary of their information. The specific function used can vary (e.g., mean, max, sum, etc.); $\mathcal{COM}^{(k)}(\cdot)$ is the update function, which takes the aggregated information and the previously hidden representation of the vertex and produces a new hidden representation for the vertex at the current layer k. This function allows the vertex to incorporate information from its neighbors into its own representation. The core of the graph neural network model consists of these elements, which allow vertices to acquire representations that encompass both the structure and features of the graph.

3 Model

3.1 Overview

The overall framework of the model is shown in Figure 2. FPGN is mainly composed of two parts, the information recording module, and the prediction module. In more detail, the role of the information recording module is to obtain the memory of each vertex and record the most recent neighbors with each vertex based on the information in a temporal bipartite graph. And relying on graph neural networks and multilayer perceptrons (MLP), and using

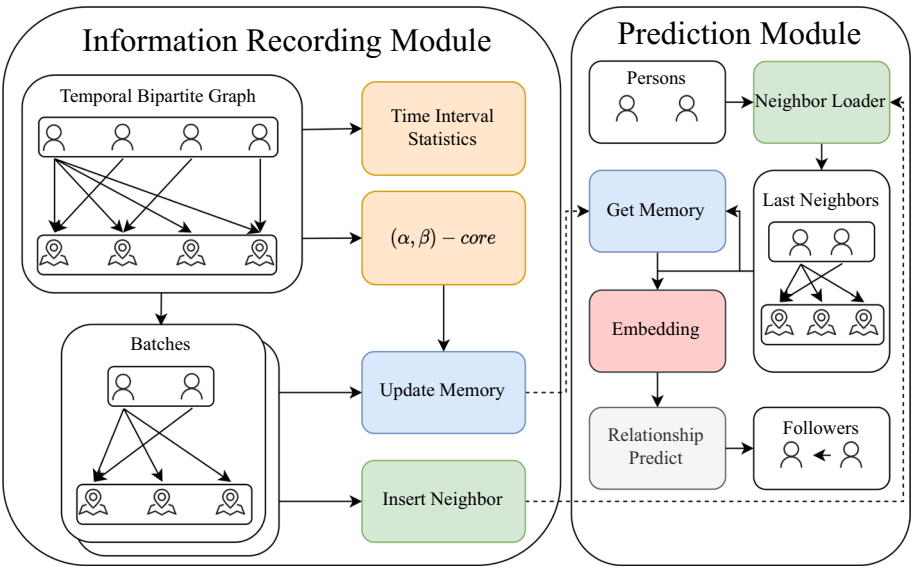

Figure 2 Overview

the memory of vertices and the most recent neighbors of these vertices, the prediction module can predict followers. The model is described in detail in this section.

3.2 Pre-processing stage

FPGN aims to solve the follower prediction problem on featureless temporal bipartite graphs. With embedding, the ID of each vertex can be mapped to a vector $\mathbf{r}_o \in \mathbf{R}_o$. We take this vector as the initial feature of the vertex. However, the features generated in this way can only help the model distinguish vertices but cannot provide more information to help the model perform better. In order to enable the model to mine better vertex representations, we also need to obtain more information in the graph.

3.2.1 Graph structure information

To be able to help the model extract information about the graph structure, we compute (α, β)-core for each vertex. More specifically, we compute whether each vertex exists with a certain set of subgraphs satisfying (α, β)-core and treat that graph structure information as a part of the vertex features. Thanks to the graph structure information, FPGN can produce more precise results. Given an upper threshold and a lower threshold for α and β, respectively: $\alpha_\tau^- \leq \alpha \leq \alpha_\tau^+$, and $\beta_\tau^- \leq \beta \leq \beta_\tau^+$, and then determine whether each vertex is within these (α, β)-core subgraphs to obtain the graph structure characteristic matrix $\mathbf{R}_s \in \mathbb{R}^{(|U|+|V|)\times d_s)}$, where $d_s = (\alpha^+ - \alpha^- + 1) \times (\beta^+ - \beta^- + 1)$ equals the number of (α, β)-core queried. Each vertex's graph structure feature $\mathbf{r}_s \in \{0, 1\}^{d_s}$ is a part of $\mathbf{R}_s$. If the value of a dimension of the feature is 0, it means that the vertex does not exist in the (α, β)-core corresponding to that dimension, and vice versa.

In the end, the characteristic matrix of the vertices used by FPGN is:

$$\mathbf{R} = \mathbf{R}_o || \mathbf{R}_s, \tag{3}$$

where $||$ is concatenation operation. Generating the vertex features by using the above way can provide rich graph structure information for FPGN while avoiding increasing its time complexity. For convenience in the description, we represent $\mathbf{R}(u)$ as a characterization of u.

3.2.2 Time interval statistics

It is crucial for the follower prediction problem to get information on the time interval between the follower's journey to a certain location and that of the leader. This information can help the model discover the following behavior and solve the follower prediction problem. Firstly, the number of times the same follower and leader travel to the same location sequentially can reveal the correlation between them. This information can be obtained statistically using the following formula:

$$\mathbf{S}_{n,(u,u')} = \sum_{v \in V} [\Delta(u, u', v) \leq \tau_t], \tag{4}$$

where $[\cdot]$ is Iverson bracket. It is a useful tool for mapping any statement to a function of the free variables in that statement. This bracket serves as a kind of truth-value indicator to assign a value of 1 to any true statement and 0 to any false statement:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

And $\mathbf{S}_{n,(u,u')}$ denotes the number of locations where the follower u' arrives to follow the leader u. $\mathbf{S}_n$ can record the frequency of follow-up between two visitors very well, which can play a significant role in the following prediction.

Then we count the closeness of the following actions for each follower, i.e., the time interval between two visitors arriving at the same location. The smaller the time interval, the closer that follower is to that leader. However, the follower can follow the leader in multiple locations. To better inform the model, we calculated the average and variance of their time intervals. Given a leader u, a follower u', We can get the average statistics and variance statistics of Δ:

$$\begin{aligned}\mathbf{S}_{tm,(u,u')} &= \sum_{v,\Delta(u,u',v)\leq\tau_t} \frac{\Delta(u,u',v)}{[\Delta(u,u',v)\leq\tau_t]},\\ \mathbf{S}_{tv,(u,u')} &= \sum_{v,\Delta(u,u',v)\leq\tau_t} \frac{(\Delta(u,u',v)-\mathbf{S}_{tm,(u,u')})^2}{[\Delta(u,u',v)\leq\tau_t]},\end{aligned} \tag{6}$$

Using this statistical information, it is possible to help the model gain a better understanding of the relationship between visitors. This information can be used to build features that highlight the relationship between leaders and followers:

$$\mathbf{S}_{(u,u')} = \text{Norm}(\mathbf{S}_{n,(u,u')})||\text{Norm}(\mathbf{S}_{tm,(u,u')})||\text{Norm}(\mathbf{S}_{tv,(u,u')}), \tag{7}$$

where Norm is the normalization operation. The features $\mathbf{S}$ obtained will help the model optimize the parameters during the training process. The details of the training process will be explained in Section 3.5.

3.3 Information update stage

Motivated by TGN, FPGN extracts memory and saves the last neighbors for each vertex so that FPGN can build the subgraph of target vertices and achieve follower prediction. However, TGN can only train for the samples in each batch separately. In follower prediction, there is no guarantee that each round of batch can provide the prediction module with features for the vertices that need to be predicted. To solve the problem, we design the module of FPGN based on TGN. More specifically, FPGN updates only the memory and neighbors of each vertex in each batch. The prediction module and back-propagation will only be done after these updates are completed.

3.3.1 Memory update

To better learn the temporal graph information, FPGN mines the vertex memory with temporal information. Given a visitor record triplet (u, v, t), which means that the visitor u arrives at the location v at timestamp t with message $\mathbf{M}_{(u,v)} = \mathbf{R}(u)||\mathbf{R}(v)$. The memory of time t for u is defined as $\mathbf{F}_u(t)$, and $\mathbf{F}_u(0) = \{0\}^{1\times d}$ where d is the dimension of the memory. The memory of u and v is calculated separately. For the sake of description, we only describe the process of updating u's memory.

To better explore the historical activity message of vertices, all activity timestamps $T_u(t)$ of u earlier than time t will be extracted and synthesized into vectors $\mathbf{T}_u(t) \in \mathbb{R}^{1\times|T_u(t)|}$. And memories $\mathbf{M}_u(t)$ of u at those times are also involved in the vertex memory update task. FPGN first encodes the activity of u:

$$\mathbf{X}_{t,u} = \cos(\mathbf{W}_t \times (\mathbf{T}_u(t) - t_u \cdot \mathbf{I}_{(t,u)}) + \mathbf{b}_t), \tag{8}$$

where $\cos(\cdot)$ is the cosine function; $\mathbf{W}_t$ is a matrix of trainable parameters and $\mathbf{b}_t$ is a trainable parameter; t_u is the timestamp of the last activity of u; $\mathbf{I}_{(t,u)} \in \mathbb{R}^{1\times|T_u(t)|}$ is a vector whose elements are all 1 with dimensions $1 \times |T_u(t)|$. It should be noted that t_u is initialized with a value of 0.

In order to extract information about the past activities of u, we need to extract all neighbors of u before time t from the adjacency matrix set A. This can be done by defining the set of neighbors of u at time t as follows:

$$\mathcal{N}_u^t = \{(v', t') | \mathbf{A}_{u,v'}^{t'} = 1, 0 \leq t' < t\}, \tag{9}$$

where $\mathbf{A}^t$ is the adjacency matrix at time t, and $\mathbf{A}_{u,v}^t = 1$ if u and v connected with each other at time t. After obtaining these neighboring vertices, we can then get the message between u and them, these neighbors' memories at that time, and the past memories of u at all timestamps before t:

$$\begin{aligned}
\mathbf{M}_{\mathcal{N}_u^t} &= \bigcup_{v',\exists(v',t')\in\mathcal{N}_u^t} \mathbf{M}_{(u,v')}, \\
\mathbf{X}_{\mathcal{N}_u^t} &= \bigcup_{v',\exists(v',t')\in\mathcal{N}_u^t} \mathbf{F}_{v'}(t'), \\
\mathbf{X}_u &= \bigcup_{t',\exists(v',t')\in\mathcal{N}_u^t} \mathbf{F}_u(t'),
\end{aligned} \tag{10}$$

then, concatenate $\mathbf{X}_u$, $\mathbf{X}_{\mathcal{N}_u^t}$ and $\mathbf{M}_{\mathcal{N}_u^t}$, as well as $\mathbf{X}_{t,u}$ to obtain the initial memory of u:

$$\mathbf{X}_{t,u}^{init} = \mathbf{X}_u || \mathbf{X}_{\mathcal{N}_u^t} || \mathbf{M}_{\mathcal{N}_u^t} || \mathbf{X}_{t,u}, \tag{11}$$

where $\mathbf{X}_{t,u}^{init}$ is the initial memory of u at timestamp t.

Because early activity records should not have too much influence on the present, and each vertex memory at every moment contains activity information before that moment, to make FPGN pay more attention to recent activity information and reduce its computational cost, we only keep the latest $\mathcal{K}$ sets of initial memory $\bar{\mathbf{X}}_u^{init}(t)$ in $\mathbf{X}_u^{init}(t)$.

LSTM is a widely used recurrent neural network architecture that is particularly suited to processing sequential data, such as speech and text [35, 38]. There are many works that handle temporal graphs using LSTM and have achieved good performance [6, 50] nowadays. However, recent research has found that the newer Gated Recurrent Unit (GRU) [10] architecture offers comparable performance to LSTM while being more computationally efficient [5, 5, 51], making it an attractive alternative for processing temporal graph datasets. Therefore, in this work, we chose to use GRU to obtain vertex memories.

To be more precise, during each time step, a GRU unit utilizes the initial memories $\bar{\mathbf{X}}_u^{init}(t)$ and the most recent memories of u, denoted as $\mathbf{F}_u(t')$, to compute the reset gate $\mathbf{p}_t$ and update gate $\mathbf{q}_t$ in the following manner:

$$\begin{aligned}
\mathbf{p}_t &= \sigma(\mathbf{W}_{px}\bar{\mathbf{X}}_u^{init}(t) + \mathbf{W}_{pf}\mathbf{F}_u(t') + \mathbf{b}_p) \\
\mathbf{q}_t &= \sigma(\mathbf{W}_{qx}\bar{\mathbf{X}}_u^{init}(t) + \mathbf{W}_{qf}\mathbf{F}_u(t') + \mathbf{b}_q),
\end{aligned} \tag{12}$$

where σ is non-linear activation functions, $\mathbf{W}_{px}$, $\mathbf{W}_{pf}$ and $\mathbf{W}_{qx}$, $\mathbf{W}_{qf}$ are weight matrices, and $\mathbf{b}_p$, $\mathbf{b}_q$ are bias vectors. The reset gate controls how much of the previous memories should be ignored, while the update gate controls how many new memories should be added to the initial memories. The candidate memory $\widetilde{\mathbf{F}}_u(t)$ is then computed as follows:

$$\widetilde{\mathbf{F}}_u(t) = \sigma(\mathbf{W}_x\bar{\mathbf{X}}_u^{init}(t) + \mathbf{W}_f(\mathbf{p}_t \odot \mathbf{F}_u(t')) + \mathbf{b}_f) \tag{13}$$

where $\odot$ denotes the element-wise product, $\mathbf{W}_x$ and $\mathbf{W}_f$ are weight matrices and $\mathbf{b}_f$ is a bias vector. Finally, the new memory of u $\mathbf{F}_u(t)$ is calculated as a linear interpolation between the previous memory and the candidate memory, controlled by the update gate:

$$\mathbf{F}_u(t) = (1 - \mathbf{q}_t) \odot \mathbf{F}_u(t') + \mathbf{q}_t \odot \widetilde{\mathbf{F}}_u(t). \tag{14}$$

GRU's mechanism allows it to process temporal graphs by selectively retaining or discarding information as time passes. This feature enables GRU to store and analyze data over an extended period, making it an effective tool for handling dynamic and evolving datasets. After completing the training of all data through the above method, we can obtain the final vertex memories:

$$\mathbf{F} = \bigcup_{n \in U \cup V} \mathbf{F}_n(\Gamma_n), \tag{15}$$

where Γ_n is the last timestamp of vertex n.

3.3.2 Neighbors update

Each round of memory update for each vertex requires tracing the vertices that were previously connected to it. Therefore, after each round of vertex memory update for a triplet (u, v, t), the edges involved in that round of update need to be saved into the adjacency matrix $\mathbf{A}^t$, that is to say, $\mathbf{A}^t_{u,v} = 1$.

Additionally, after acquiring $\mathbf{F}$, utilizing all the connection information for improved follower predictions is imperative. Thus, amalgamating all the adjacency matrices from each time step into a single matrix becomes necessary:

$$\mathbf{A}_a = \max_{t=1}^{T} \{\mathbf{A}^t\} \tag{16}$$

3.4 Prediction module

Given a visitor u, by utilizing $\mathbf{A}_a$, we can extract all the locations u have visited, represented as $\mathcal{N}_a(u)$. It should be noted that to ensure FPGN can retain the memory of each vertex during the learning process, vertex u will be included in $\mathcal{N}_a(u)$. Then, FPGN can use the memories of these vertices (e.g., the memory of u is $\mathbf{F}_u$) as their features to further learn about the characteristics of these two visitors through GNNs.

In FPGN, we use Graph Attention Network (GAT) [42] for vertex representation learning. GAT is a multi-head attention mechanism-based GNN model that enables efficient and effective vertex and edge relationship modeling in graph-structured data. One of the key advantages of the GAT model is that it can learn to dynamically assign different weights to different vertices and edges in the graph while considering the neighborhood and global information. This is achieved using a self-attention mechanism that allows the model to selectively attend to different subsets of vertices and edges based on their relevance to the target vertex or subgraph. With the attention mechanism of GAT, FPGN can better determine which locations are more important for determining the relationship between two visitors. Moreover, using multiple attention heads allows the model to learn diverse representations of the graph, thereby improving the robustness and generalization capabilities of the model.

GAT in FPGN can be mathematically formulated as follows: Given a visitor vertex u with memory $\mathbf{F}_u$, and the locations $\mathcal{N}_a(u)$ visited by u, the output features of the l-th layer can be

computed as:

$$\mathbf{h}_u^{(l)} = \|_{k=1}^{K} \sigma\left([\sum_{v \in \mathcal{N}_a(u)} a_{k,uv}^{(l)} \mathbf{W}_k^{(l)} \mathbf{h}_v^{(l-1)}] \right), \tag{17}$$

where K is the number of attention heads, σ is a non-linear activation function (e.g., ReLU), $\mathbf{W}_k^{(l)}$ is the weight matrix for the k-th attention head at the l-th layer, and $a_{k,uv}^{(l)}$ is the attention weight assigned to the edge from vertex u to vertex v by the k-th attention head at l-th layer, $\mathbf{h}_u^{(l)}$ is l-th layer representation of u, and $\mathbf{h}_u^{(0)} = \mathbf{F}_u$, $\mathbf{h}_v^{(0)} = \mathbf{F}_v$.

Follow prediction problem involves bipartite graph datasets. However, the conventional GAT model is inefficient in distinguishing between different types of vertices on the bipartite graph and cannot handle edge messages. To overcome these limitations, we drew inspiration from UniMP [37] and mapped different types of vertices differently while utilizing edge information. More specifically, for the c-th head in layer l, the representations of u, v, and their edge, as well as the attention weight between them, can be calculated using the following method:

$$\begin{aligned}
\mathbf{h}_{c,u}^{(l)} &= \mathbf{W}_{c,u}^{(l)} \mathbf{h}_u^{(l)} + \mathbf{b}_{c,u} \\
\mathbf{h}_{c,v}^{(l)} &= \mathbf{W}_{c,v}^{(l)} \mathbf{h}_v^{(l)} + \mathbf{b}_{c,v} \\
\mathbf{h}_{c,uv} &= \mathbf{W}_{c,m}^{(l)} \mathbf{h}_{m,uv} + \mathbf{b}_{c,m} \\
a_{c,uv}^{(l)} &= \frac{< \mathbf{h}_{c,u}^{(l)}, \mathbf{h}_{c,v}^{(l)} + \mathbf{h}_{c,uv} >}{\sum_{v' \in \mathcal{N}_a(u)} < \mathbf{h}_{c,u}^{(l)}, \mathbf{h}_{c,v'}^{(l)} + \mathbf{h}_{c,uv'} >},
\end{aligned} \tag{18}$$

where $a_{c,uv}^{(l)}$ is the new attention weight, $< \mathbf{h}_1, \mathbf{h}_2 >= \exp(\frac{\mathbf{h}_1^T \mathbf{h}_2}{\sqrt{d_h}})$ is exponential scale dot-product function and d_h is the hidden size of each head, $\mathbf{W}_{c,u}^{(l)}$, $\mathbf{W}_{c,v}^{(l)}$, and $\mathbf{W}_{c,m}^{(l)}$ represent weight matrices, while $\mathbf{b}_{c,u}$, $\mathbf{b}_{c,v}$, and $\mathbf{b}_{c,m}$ are bias vectors at the l-th layer. Additionally, the variables $\mathbf{h}_{c,u}^{(l)}$ and $\mathbf{h}_{c,v}^{(l)}$ represent the l-th layer representations of u, v; $\mathbf{h}_{m,uv}$ and $\mathbf{h}_{c,uv}$ represent the message and hidden representation of their edge. It is important to note that for the initial layer, we have $\mathbf{h}_{c,u}^{(0)} = \mathbf{F}_u$, $\mathbf{h}_{c,v}^{(0)} = \mathbf{F}_v$, and each layer of $\mathbf{h}_{m,uv}$ can be generated as follows:

$$\mathbf{h}_{m,uv} = \cos(\mathbf{W}_t \times (t - t_u) + \mathbf{b}_t) || \mathbf{M}_{(u,v)}, \tag{19}$$

where timestamp t indicates the occurrence of the current connection, allowing the model to leverage user activity information.

After the above adjustments, corresponding adjustments also need to be made to the mapping function for different types of vertices. For example, the process of mapping u is as follows:

$$\begin{aligned}
\mathbf{h}_{\eta,u}^{(l)} &= \mathbf{W}_{\eta,u}^{(l-1)} \mathbf{h}_u^{(l-1)} + \mathbf{b}_{\eta,u}^{(l)} \\
\mathbf{h}_{\eta,v}^{(l)} &= \mathbf{W}_{\eta,v}^{(l-1)} \mathbf{h}_v^{(l-1)} + \mathbf{b}_{\eta,v}^{(l)} \\
\hat{\mathbf{h}}_u^{(l)} &= \|_{c=1}^{C} [\sum_{v \in \mathcal{N}_a(u)} a_{c,uv}^{(l)} \mathbf{W}_c^{(l)} \mathbf{h}_v^{(l-1)}] \\
\widetilde{\mathbf{h}}_u^{(l)} &= \sigma(\mathbf{W}_g^{(l)} (\hat{\mathbf{h}}_u^{(l)} || \mathbf{h}_{\eta,u}^{(l)} || (\hat{\mathbf{h}}_u^{(l)} - \mathbf{h}_{\eta,u}^{(l)}))) \\
\mathbf{h}_u^{(l)} &= \sigma(\text{LayerNorm}((1 - \widetilde{\mathbf{h}}_u^{(l)}) \hat{\mathbf{h}}_u^{(l)} + \widetilde{\mathbf{h}}_u^{(l)} \mathbf{h}_{\eta,u}^{(l)})),
\end{aligned} \tag{20}$$

where $\mathbf{W}_{\eta,u}^{(l)}$, $\mathbf{W}_{\eta,v}^{(l)}$, $\mathbf{W}_c^{(l)}$, $\mathbf{W}_g^{(l)}$ are weight matrices and $\mathbf{b}_{\eta,u}^{(l)}$, $\mathbf{b}_{\eta,v}^{(l)}$ are bias vectors, LayerNorm$(\cdot)$ is layer normalization function.

Given two visitor vertices u and u', their memories are processed through GAT, and the final output of the last layer $\mathbf{h}_u^{(L)}$ and $\mathbf{h}_{u'}^{(L)}$ can be used to predict their relationship with the help of the following module:

$$\begin{aligned} \mathbf{h}_{uv}^{p} &= \sigma(\mathbf{W}_{p_1}(\mathbf{h}_u^{(L)}) + \mathbf{b}_{p_1} + \mathbf{W}_{p_2}(\mathbf{h}_{u'}^{(L)}) + \mathbf{b}_{p_2}) \\ \hat{y}_{u,u'} &= \sigma(\mathbf{W}_y \mathbf{h}_{uv}^{p} + \mathbf{b}_y), \end{aligned} \tag{21}$$

where $\mathbf{W}_{p_1}, \mathbf{W}_{p_2}, \mathbf{W}_y$ are weight matrices and $\mathbf{b}_{p_1}, \mathbf{b}_{p_2}, \mathbf{b}_y$ are bias vectors, $\hat{y}_{u,u'} \in [0, 1]$ is the probability that u' follows u predicted by FPGN.

3.5 Training objective

The follower prediction problem can be approached as a binary classification task. As such, the model can be trained using binary cross-entropy as the loss function. The binary cross-entropy loss function is commonly used in machine learning to train models for binary classification tasks. It measures the difference between the predicted and ground truth by using the logarithmic loss. The formula for it can be written as follows:

$$\mathcal{L}_m = \frac{1}{|D_{train}|} \sum_{(u1,u2)\in D_{train}} y_{u,u'} \log \hat{y}_{u,u'} + (1 - y_{u,u'}) \log(1 - \hat{y}_{u,u'}) \tag{22}$$

where D_{train} is the training set, $|D_{train}|$ is the size of training set, $y_{u,u'}$ represents the ground truth indicating whether u' is a follower of u. Specifically, if the result is positive, then $y_{u,u'} = 1$; otherwise, it equals 0. $\hat{y}_{u,u'}$ refers to the predicted probability of a positive outcome. $y_{u,u'} \log \hat{y}_{u,u'}$ heavily penalizes misclassification of positive instances by the model. Similarly, $(1 - y_{u,u-2}) \log(1 - \hat{y}_{u-11_u})$ penalizes misclassification of negative instances. Moreover, it is evident that the loss function necessitates the predicted values to be within the range of 0 and 1, denoted as $0 \leq \hat{y}_{u,u'} \leq 1$. As a result, before feeding into this loss function, the predicted outcomes will undergo a Sigmoid transformation.

Furthermore, as explained in Section 3.2.2, time interval statistics are also important information that helps the model solve the follower prediction problem. Therefore, FPGN also uses the following functions during training:

$$\begin{aligned} \mathbf{h}_{uv}^{s} &= \sigma(\mathbf{W}_{s_1}(\mathbf{h}_u^{(L)}) + \mathbf{b}_{s_1} + \mathbf{W}_{s_2}(\mathbf{h}_{u'}^{(L)}) + \mathbf{b}_{s_2}) \\ \hat{y}_{u,u'}^{s} &= \sigma(\mathbf{W}_{s_3} \mathbf{h}_{uv}^{s} + \mathbf{b}_{s_3}) \\ \mathcal{L}_s &= \frac{1}{|D_{train}|} \sum_{(u,u')\in D_{train}} ||\hat{y}_{u,u'}^{s} - \mathbf{S}_{u,u'}||_2^2, \end{aligned} \tag{23}$$

where $\mathbf{W}_{s_1}, \mathbf{W}_{s_2}, \mathbf{W}_{s_3}$ are weight matrices and $\mathbf{b}_{s_1}, \mathbf{b}_{s_2}, \mathbf{b}_{s_3}$ are bias vectors, $||\cdot||_2^2$ representing L2 norm.

The final loss function can be represented as:

$$\mathcal{L} = (1 - \lambda) \times \mathcal{L}_m + \lambda \times \mathcal{L}_c, \tag{24}$$

where λ is a hyperparameter used to adjust the weights of two sets of loss functions. Since $\mathcal{L}_m$ is the main loss function of FPGN, while $\mathcal{L}_c$ mainly plays a supporting role, therefore in actual training, λ will be adjusted to a relatively small value.

4 Experiment

This section first introduces the experimental setup, then explains the method of generating the dataset, followed by an introduction to the baselines compared with FPGN, and finally presents a large number of experimental results. These results demonstrate the excellent performance of FPGN. In addition, this section also shows the results of ablation experiments, proving that each part of FPGN plays a positive role in improving model accuracy.

4.1 Experimental setup

In all precision experiments, the hyperparameters of FPGN remain the same. Among them, epoch is 340, learning rate is 1×10^{-4}, batch size is 1×10^4, the number of selected latest initial memory sets in $\mathbf{X}_u^{init}(t)$ is $\mathcal{K} = 10$, the number of heads in GAT is $C = 2$, vertex ID embedding dimension is 32, $\alpha_\tau^- = \alpha_\tau^+ = 2$, $\beta_\tau^- = 1$, $\beta_\tau^+ = 20$, all module hidden layer dimensions are 64, loss function weight $\lambda = 0.05$ and Adam optimizer is chosen. The algorithm we choose to implement (α, β)-core is proposed by work [23]. In order to improve the efficiency of the model, we use the source code provided by the authors [1], and embed it into Python code using SWIG [2]. The experiment is conducted on a server with two Intel(R) Xeon(R) Silver 4208 CPUs, one NVIDIA RTX 6000, and 64GB of memory.

In our experiment, we utilize two widely adopted evaluation metrics of edge prediction on temporal bipartite graphs, namely Average Precision (AP) and AUC, as the metrics of the follower prediction problem. The definitions of these metrics are as follows:

$$\begin{aligned} AP &= \sum_{n=1}^{N} P(n)\Delta R(n), \\ P &= \frac{TP}{TP + FP}, \\ R &= \frac{TP}{TP + FN}, \\ AUC &= \frac{\sum_{i=1}^{m} rank_i - \frac{m(m+1)}{2}}{n_1 \times n_0}. \end{aligned} \tag{25}$$

4.1.1 Dataset

To the best of our knowledge, there is currently no open-source dataset available for predicting followers. To address this gap, we utilized the widely-used temporal bipartite graph datasets named Wikipedia and Reddit [16, 21, 33, 47] to create our own dataset. The statistical details of these two datasets are presented in Table 1.

In the follower prediction problem, we consider all connections to only have the meaning of "visit," so the dataset we use does not include edge features. We use 80% of the data in each dataset as the training set and the remaining 20% as the test set. Additionally, 20% of the training set is treated as a validation set. It should be noted that in order to ensure that the model does not test visitors who have not seen before, we remove visitors that only appear in the test set.

[1] https://github.com/boge-liu/alpha-beta-core

[2] https://github.com/swig/

Table 1 Dataset Statistics

	$\|U\|$	$\|V\|$	# Edge
Wikipedia	8,227	1,000	157,474
Reddit	10,000	984	672,358

To generate ground truth, we set a fixed ratio time window. Specifically, assuming the time span of the dataset is Γ, two visitors u and u', and a location v, we consider u and u' are leader and follower when $\Delta(u, u', v) \leq \Gamma \times 10^{-4}$. The negative samples are generated from samples other than the positive samples, and the number of negative samples is twice that of the positive samples. In the Wikipedia dataset, the links represent users' edits to Wikipedia pages. The obtained Wikidata dataset can help FPGN predict those who follow other editors in editing Wikipedia pages. Reddit contains numerous user interaction information in numerous posts. The Reddit dataset obtained using the above method allows FPGN to predict followers who engage in reply conversations with specific users.

4.1.2 Compared methods

To the best of our knowledge, there is currently no model specifically optimized for the follower prediction problem. Therefore, we use six widely used GNN models: GCN [19], GAT [42], GE [53], SAGE [15], GNN-FiLM [4] and GIN [49] as baselines, where GE is the general GNN layer adapted from [53]. All of these algorithms have the structure shown in (2), but differ in the way they aggregate and update the information.

In addition, since the edge prediction problem on temporal graphs is similar to the follower prediction problem, we also compare with state-of-the-art algorithms JODIE [20] and NeurTW [16]. JODIE employs two recurrent neural networks to update a visitor's and a location's embedding at every interaction. In addition, this model has a novel projection operator that learns to estimate the embedding of the visitor at any time in the future. And to make the method scalable, a t-Batch algorithm is developed that creates time-consistent batches and leads to faster training. NeurTW conduct spatiotemporal-biased random walks to retrieve a set of representative motifs, enabling temporal vertices to be characterized effectively. With a component based on neural ordinary differential equations, the extracted motifs allow for irregularly sampled temporal vertices to be embedded explicitly over multiple different interaction time intervals. To enrich the supervision signals, a harder contrastive pretext task is designed for model optimization. It should be noted that TGN is also a state-of-the-art algorithm for solving edge prediction problems on temporal graphs. However, as explained in Section 1, TGN cannot be directly applied to the follower prediction problem. Therefore, we do not compare TGN in accuracy evaluation experiments. Instead, we analyze the effectiveness of optimizing TGN for the follower prediction problem in ablation experiments and show that TGN does not outperform FPGN on this task.

4.2 Prediction accuracy evaluation

Table 2 shows the results of the accuracy experiments. From the table, we observe that FPGN outperforms all baseline models in terms of both AP and AUC on both datasets. Specifically, the AP of FPGN is 0.4620 on Wikipedia and 0.5457 on Reddit, which are the highest scores for both datasets. FIL achieves the second-highest AP score of 0.4180 on Wikipedia, while

Table 2 Effectiveness Evaluation Results

	Wikipedia		Reddit	
	AP	AUC	AP	AUC
GCN	0.3894	0.5491	0.3586	0.5320
GAT	0.4005	0.5549	0.3663	0.5403
GE	0.4050	0.5614	0.3689	0.5420
SAGE	0.3809	0.5331	0.3478	0.5202
GNN-FiLM	0.4180	0.5646	0.3641	0.5398
GIN	0.3957	0.5452	0.3717	0.5283
JODIE	0.3609	0.5397	N/A	N/A
NeurTW	0.3559	0.5325	N/A	N/A
FPGN	0.4620	0.6250	0.5457	0.7305

GIN obtains the second-highest AP score of 0.3717 on Reddit. In contrast, FPGN was able to improve by 10.53% on Wikipedia based on FIL, while on Reddit, it improved by 46.81% based on GIN. Similarly, on Wikipedia and Reddit datasets, FPGN outperforms other models with the highest AUC scores of 0.6250 and 0.7305, respectively. The second-best AUC score on Wikipedia is achieved by FIL at 0.5646, while GE obtains the second-best AUC score on Reddit at 0.5403. Compared to them, FPGN respectively increased by 10.70% and 35.20% based on their performance.

These results indicate that, on the one hand, because these six general GNN baseline models (i.e., GCN, GAT, GE, GraphSAGE, FIL and GIN) are designed for general graph structures, they do not perform as well as FPGN, which is specifically optimized for the follower prediction problem. This demonstrates the importance of problem-specific optimization in graph-based models. On the other hand, compared with two state-of-the-art temporal graph edge prediction models (i.e., JODIE and NeurTW), FPGN still achieves better performance. Due to the heavy emphasis on temporal information processing in these two baselines, their efficiency is very low. In experiments conducted on the Reddit dataset, both JODIE and NeurTW were unable to obtain experimental results within 48 hours. In addition, because they are unable to fully utilize the time window information during follower and leader activity processes, they are more easily misled and achieve worse results than general GNN models. More specifically, the architecture of JODIE and NeurTW makes them more suitable for utilizing historical information to predict whether a particular event will occur. However, when it comes to predicting the time difference of future events, their method of handling timestamp information has a negative effect on prediction.

In summary, the proposed FPGN model is highly effective in predicting followers on both datasets. It outperforms six general GNN models and two state-of-the-art temporal graph edge prediction models. These results demonstrate the exceptional performance of FPGN on this task.

4.3 Ablation experiment

In order to better achieve the follower prediction task, FPGN strengthened the utilization of time interval statistics information and graph structure information. To demonstrate the advanced nature of these optimizations, we conducted ablation experiments. Specifically, we compared FPGN with only improved TGN (referred to as FGN), FPGN without graph

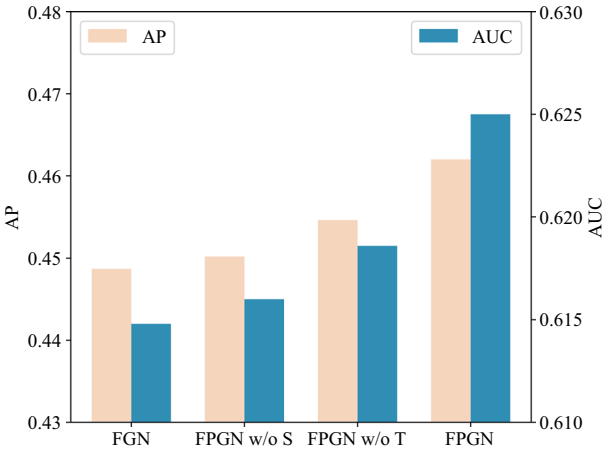

Figure 3 Ablation Experimental Results

structure information (referred to as FPGN w/o S) and FPGN without time interval statistics information (referred to as FPGN w/o T). The experimental results on Wikipedia are shown in Figure 3.

After analyzing the experimental results, it can be found that both the AP and AUC metrics show that FGN is the worst, and FPGN is the best. This is because TGN can only make good use of the updated information of each vertex and fully consider the information at different times for the same vertex, but cannot effectively utilize time information between two different vertices. Therefore, although FGN performs better in follower prediction compared to JODIE and NeurTW, its performance is still worse than FPGN as well as FPGN w/o S and FPGN w/o T.

By comparing the effects of FPGN w/o S and FPGN w/o T, it can be found that the accuracy of FPGN w/o T is higher than that of FPGN w/o S. This is because (α, β)-core can extract the activity information of each vertex very well. There is no doubt that more active vertices are more likely to exhibit the following behavior. Therefore, the information extracted by (α, β)-core can better help the model predict future following behavior.

5 Related works

Graph Neural Networks Graph Neural Networks (GNNs) have gained significant attention in recent years due to their ability to represent and learn from graphs and other information [4, 17, 22, 25, 48]. One of the earliest works in the development of GNNs is the Graph Convolutional Network (GCN) [19]. GCN is a type of neural network that operates directly on graphs, allowing it to learn representations that capture the underlying structure of the graph. Another notable work is Graph Attention Networks (GAT) [42]. GAT uses attention mechanisms to weigh the importance of each vertex's neighbors, allowing the network to focus on the most relevant information for each vertex. Other related works in the field of GNNs include GraphSAGE [15], GIN [49], *etc*. These approaches have been used in various applications such as social network analysis, protein function prediction, and knowledge graph completion [43]. In addition, there are some GNN models that have been specifically

optimized for bipartite graph problems [54, 56, 57]. However, these methods cannot utilize temporal information and are therefore not suitable for application on temporal bipartite graphs.

GNNs on Temporal Graphs In recent years, graph neural network models specifically designed for temporal graphs have been continuously emerging. Early models for temporal graphs focused on DTDGs. There are two encoding methods for DTDGs: one is to aggregate graph snapshots and apply static methods [13, 34, 36, 59], and the other is to modify the behavior of subsequent snapshots through random walks [12, 29, 46]. Based on CTDGs, several methods have recently been proposed for temporal graphs, including random walk models [2, 30] and sequence-based methods [20, 28, 39]. What's more, many temporal graph architectures focus on updating vertex-wise memory when new interactions occur [3, 20, 33, 40, 47]. These algorithms can fully utilize the connectivity and temporal information and perform better. However, there is great confusion in predicting follower problems for these methods because it is difficult to distinguish the order of vertex activity and time difference. This makes these algorithms even misled by temporal information when making follower predictions, resulting in low prediction accuracy.

Cohesive Subgraph Mining Many algorithms are now available for finding dense subgraphs, such as clique [8, 9], quasi-clique [41], k-core [7, 18], and k-truss [11, 44], *etc*. At the same time, there are also many methods specifically designed to search for dense subgraphs on bipartite graphs, such as biclique [1, 58], k-bitruss [45], bi-triangle [52], (α, β)-core [23], δ-quasi-biclique [24] and k-biplex [55], *etc*.

These algorithms cannot utilize temporal information or directly implement edge classification, so they cannot be directly used to solve the follower prediction problem. However, followers often exist in denser subgraphs, so these algorithms can provide good auxiliary support for solving this problem.

6 Conclusion

This paper introduces a novel approach to solving the problem of follower prediction in temporal bipartite graphs using a model called Follower Prediction Graph Network (FPGN). The model is designed to accurately predict high-risk visitors by tracking their activities in the same location as infected visitors. FPGN is inspired by the state-of-the-art temporal graph edge prediction algorithm TGN and addresses its shortcomings. The high accuracy of the FPGN is attributed to its well-designed approach to addressing this problem, including making full use of time windows and global structure information. The results of extensive experiments conducted on two real datasets demonstrate the effectiveness of FPGN. In conclusion, this work represents a significant advancement in the problem of follower prediction and provides a new and effective approach to solving this challenging problem.

Author Contributions Jianke Yu, Xianhang Zhang, Hanchen Wang, and Xiaoyang Wang wrote the main manuscript text. Wenjie Zhang and Ying Zhang provided key ideas of the proposed model. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions The paper was supported by ZJNSF LY21F020012.

Availability of data and materials The code and data are available at https://github.com/yujianke100/FPGN.

Declarations

Ethical Approval Not applicable since there are no human and/ or animal studies included in this paper.

Competing Interests No, we declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

1. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P.L., Simeone, B.: Consensus algorithms for the generation of all maximal bicliques. Discret. Appl. Math. **145**(1), 11–21 (2004). https://doi.org/10.1016/j.dam.2003.09.004
2. Bastas, N., Semertzidis, T., Axenopoulos, A., Daras, P.: evolve2vec: Learning network representations using temporal unfolding. In: Kompatsiaris, I., Huet, B., Mezaris, V., Gurrin, C., Cheng, W., Vrochidis, S. (eds.) MultiMedia Modeling - 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8-11, 2019, Proceedings, Part I, Lecture Notes in Computer Science, vol. 11295, pp. 447–458. Springer (2019). https://doi.org/10.1007/978-3-030-05710-7_37
3. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V.F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H.F., Ballard, A.J., Gilmer, J., Dahl, G.E., Vaswani, A., Allen, K.R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M.M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. CoRR (2018) arXiv:1806.01261
4. Brockschmidt, M.: Gnn-film: Graph neural networks with feature-wise linear modulation. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event, Proceedings of Machine Learning Research, vol. 119, pp. 1144–1152. PMLR (2020) . http://proceedings.mlr.press/v119/brockschmidt20a.html
5. Cahuantzi, R., Chen, X., Güttel, S.: A comparison of LSTM and GRU networks for learning symbolic sequences. CoRR (2021). arXiv:2107.02248
6. Chen, G., Hu, L., Zhang, Q., Ren, Z., Gao, X., Cheng, J.: ST-LSTM: spatio-temporal graph based long short-term memory network for vehicle trajectory prediction. In: IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020, pp. 608–612. IEEE (2020). https://doi.org/10.1109/ICIP40778.2020.9191332
7. Cheng, J., Ke, Y., Chu, S., Özsu, M.T.: Efficient core decomposition in massive networks. In: 2011 IEEE 27th International Conference on Data Engineering, pp. 51–62. IEEE (2011)
8. Cheng, J., Ke, Y., Fu, A.W., Yu, J.X., Zhu, L.: Finding maximal cliques in massive networks. ACM Trans. Database Syst. **36**(4), 21:1-21:34 (2011). https://doi.org/10.1145/2043652.2043654
9. Cheng, J., Zhu, L., Ke, Y., Chu, S.: Fast algorithms for maximal clique enumeration with limited memory. In: Yang, Q., Agarwal, D., Pei, J. (eds.) The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012, pp. 1240–1248. ACM (2012). https://doi.org/10.1145/2339530.2339724
10. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Moschitti,

A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 1724–1734. ACL (2014). https://doi.org/10.3115/v1/d14-1179
11. Cohen, J.: Trusses: Cohesive subgraphs for social network analysis. National security agency technical report **16**(3.1) (2008)
12. Du, L., Wang, Y., Song, G., Lu, Z., Wang, J.: Dynamic network embedding : An extended approach for skip-gram based network embedding. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden, pp. 2086–2092. ijcai.org (2018). https://doi.org/10.24963/ijcai.2018/288
13. Fard, A.M., Bagheri, E., Wang, K.: Relationship prediction in dynamic heterogeneous information networks. In: Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D. (eds.) Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I, Lecture Notes in Computer Science, vol. 11437, pp. 19–34. Springer (2019). https://doi.org/10.1007/978-3-030-15712-8_2
14. Gu, L., Mukherjee, M., Guo, M., Lloret, J., Matam, R.: Low-cost assistive body temperature screening system to combat communicable infectious diseases leveraging edge computing and long-range and low-power wireless networks. IEEE Internet Things J. **10**(5), 4174–4183 (2023). https://doi.org/10.1109/JIOT.2022.3215484
15. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, pp. 1024–1034 (2017). https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html
16. Jin, M., Li, Y.F., Pan, S.: Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In: Advances in Neural Information Processing Systems (2022)
17. Jin, Y., Ji, W., Shi, Y., Wang, X., Yang, X.: Meta-path guided graph attention network for explainable herb recommendation. Health Inf. Sci. Syst. **11**(1), 5 (2023). https://doi.org/10.1007/s13755-022-00207-6
18. Khaouid, W., Barsky, M., Srinivasan, V., Thomo, A.: K-core decomposition of large networks on a single pc. Proceedings of the VLDB Endowment **9**(1), 13–23 (2015)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=SJU4ayYgl
20. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (eds.) Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4–8, 2019, pp. 1269–1278. ACM (2019). https://doi.org/10.1145/3292500.3330895
21. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (eds.) Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4–8, 2019, pp. 1269–1278. ACM (2019). https://doi.org/10.1145/3292500.3330895
22. Li, C., Sun, L., Peng, D., Subramani, S., Nicolas, S.C.: A multi-label classification system for anomaly classification in electrocardiogram. Health Inf. Sci. Syst. **10**(1), 19 (2022). https://doi.org/10.1007/s13755-022-00192-w
23. Liu, B., Yuan, L., Lin, X., Qin, L., Zhang, W., Zhou, J.: Efficient (a,β)-core computation: an index-based approach. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13–17, 2019, pp. 1130–1141. ACM (2019). https://doi.org/10.1145/3308558.3313522
24. Liu, X., Li, J., Wang, L.: Quasi-bicliques: Complexity and binding pairs. In: International Computing and Combinatorics Conference, pp. 255–264. Springer (2008)
25. Lu, H., Uddin, S.: Embedding-based link predictions to explore latent comorbidity of chronic diseases. Health Inf. Sci. Syst. **11**(1), 2 (2023). https://doi.org/10.1007/s13755-022-00206-7
26. Lu, Y., Phillips, C.A., Langston, M.A.: Biclique: an r package for maximal biclique enumeration in bipartite graphs. BMC Research Notes **13**(1), 1–5 (2020)
27. Lyu, B., Qin, L., Lin, X., Zhang, Y., Qian, Z., Zhou, J.: Maximum and top-k diversified biclique search at scale. VLDB J. **31**(6), 1365–1389 (2022). https://doi.org/10.1007/s00778-021-00681-6
28. Ma, Y., Guo, Z., Ren, Z., Tang, J., Yin, D.: Streaming graph neural networks. In: Huang, J.X., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y. (eds.) Proceedings of the 43rd International ACM

SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25–30, 2020, pp. 719–728. ACM (2020). https://doi.org/10.1145/3397271.3401092
29. Mahdavi, S., Khoshraftar, S., An, A.: dynnode2vec: Scalable dynamic network embedding. In: Abe, N., Liu, H., Pu, C., Hu, X., Ahmed, N.K., Qiao, M., Song, Y., Kossmann, D., Liu, B., Lee, K., Tang, J., He, J., Saltz, J.S. (eds.) IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10–13, 2018, pp. 3762–3765. IEEE (2018). https://doi.org/10.1109/BigData.2018.8621910
30. Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Continuous-time dynamic network embeddings. In: Champin, P., Gandon, F., Lalmas, M., Ipeirotis, P.G. (eds.) Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23–27, 2018, pp. 969–976. ACM (2018). https://doi.org/10.1145/3184558.3191526
31. Peeters, R.: The maximum edge biclique problem is np-complete. Discret. Appl. Math. **131**(3), 651–654 (2003). https://doi.org/10.1016/S0166-218X(03)00333-0
32. Podder, P., Das, S.R., Mondal, M.R.H., Bharati, S., Maliha, A., Hasan, M.J., Piltan, F.: Lddnet: A deep learning framework for the diagnosis of infectious lung diseases. Sensors **23**(1), 480 (2023). https://doi.org/10.3390/s23010480
33. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.M.: Temporal graph networks for deep learning on dynamic graphs. CoRR (2020). arXiv:2006.10637
34. Sajadmanesh, S., Bazargani, S., Zhang, J., Rabiee, H.R.: Continuous-time relationship prediction in dynamic heterogeneous information networks. ACM Trans. Knowl. Discov. Data **13**(4), 44:1-44:31 (2019). https://doi.org/10.1145/3333028
35. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. CoRR (2014). arXiv:1402.1128
36. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: Caverlee, J., Hu, X.B., Lalmas, M., Wang, W. (eds.) WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3–7, 2020, pp. 519–527. ACM (2020). https://doi.org/10.1145/3336191.3371845
37. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19–27 August 2021, pp. 1548–1554. ijcai.org (2021). https://doi.org/10.24963/ijcai.2021/214
38. Staudemeyer, R.C., Morris, E.R.: Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. CoRR (2019). arXiv:1909.09586
39. Trivedi, R., Dai, H., Wang, Y., Song, L.: Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, Proceedings of Machine Learning Research, vol. 70, pp. 3462–3471. PMLR (2017). http://proceedings.mlr.press/v70/trivedi17a.html
40. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net (2019). https://openreview.net/forum?id=HyePrhR5KX
41. Tsourakakis, C.E., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.A.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: Dhillon, I.S., Koren, Y., Ghani, R., Senator, T.E., Bradley, P., Parekh, R., He, J., Grossman, R.L., Uthurusamy, R. (eds.) The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11–14, 2013, pp. 104–112. ACM (2013). https://doi.org/10.1145/2487575.2487645
42. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings. OpenReview.net (2018). https://openreview.net/forum?id=rJXMpikCZ
43. Wang, H., Yu, J., Wang, X., Chen, C., Zhang, W., Lin, X.: Neural similarity search on supergraph containment. IEEE Transactions on Knowledge and Data Engineering (2023)
44. Wang, J., Cheng, J.: Truss decomposition in massive networks. Proc. VLDB Endow. 5(9), 812–823 (2012). https://doi.org/10.14778/2311906.2311909, http://vldb.org/pvldb/vol5/p812_jiawang_vldb2012.pdf
45. Wang, K., Lin, X., Qin, L., Zhang, W., Zhang, Y.: Efficient bitruss decomposition for large-scale bipartite graphs. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 661–672. IEEE (2020)
46. Winter, S.D., Decuypere, T., Mitrovic, S., Baesens, B., Weerdt, J.D.: Combining temporal aspects of dynamic networks with node2vec for a more efficient dynamic link prediction. In: Brandes, U., Reddy, C., Tagarelli, A. (eds.) IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018, Barcelona, Spain, August 28–31, 2018, pp. 1234–1241. IEEE Computer Society (2018). https://doi.org/10.1109/ASONAM.2018.8508272

47. Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. OpenReview.net (2020). https://openreview.net/forum?id=rJeW1yHYwH
48. Xu, H., Chen, X., Qian, P., Li, F.: A two-stage segmentation of sublingual veins based on compact fully convolutional networks for traditional chinese medicine images. Health Inf. Sci. Syst. **11**(1), 19 (2023). https://doi.org/10.1007/s13755-023-00214-1
49. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net (2019). https://openreview.net/forum?id=ryGs6iA5Km
50. Xu, M., Singh, A.V., Karniadakis, G.E.: Dyng2g: An efficient stochastic graph embedding method for temporal graphs. CoRR (2021). arXiv:2109.13441
51. Yang, S., Yu, X., Zhou, Y.: Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In: 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), pp. 98–101 (2020). https://doi.org/10.1109/IWECAI50956.2020.00027
52. Yang, Y., Fang, Y., Orlowska, M.E., Zhang, W., Lin, X.: Efficient bi-triangle counting for large bipartite networks. Proceedings of the VLDB Endowment **14**(6), 984–996 (2021)
53. You, J., Ying, Z., Leskovec, J.: Design space for graph neural networks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/c5c3d4fe6b2cc463c7d7ecba17cc9de7-Abstract.html
54. Yu, J., Wang, H., Wang, X., Li, Z., Qin, L., Zhang, W., Liao, J., Zhang, Y.: Group-based fraud detection network on e-commerce platforms. In: Singh, A., Sun, Y., Akoglu, L., Gunopulos, D., Yan, X., Kumar, R., Ozcan, F., Ye, J. (eds.) Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6–10, 2023, pp. 5463–5475. ACM (2023). https://doi.org/10.1145/3580305.3599836
55. Yu, K., Long, C., Liu, S., Yan, D.: Efficient algorithms for maximal k-biplex enumeration. In: Proceedings of the 2022 International Conference on Management of Data, pp. 860–873 (2022)
56. Zhang, X., Wang, H., Yu, J., Chen, C., Wang, X., Zhang, W.: Polarity-based graph neural network for sign prediction in signed bipartite graphs. World Wide Web **25**(2), 471–487 (2022). https://doi.org/10.1007/s11280-022-01015-4
57. Zhang, X., Wang, H., Yu, J., Chen, C., Wang, X., Zhang, W.: Bipartite graph capsule network. World Wide Web (WWW) **26**(1), 421–440 (2023). https://doi.org/10.1007/s11280-022-01009-2
58. Zhang, Y., Phillips, C.A., Rogers, G.L., Baker, E.J., Chesler, E.J., Langston, M.A.: On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. BMC Bioinform. **15**, 110 (2014). https://doi.org/10.1186/1471-2105-15-110
59. Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., Cai, D.: What to do next: Modeling user behaviors by time-lstm. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19–25, 2017, pp. 3602–3608. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/504