Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: www.tandfonline.com/journals/uaai20

Friend Link Prediction Method Based on Heterogeneous Multigraph and Hierarchical Attention

Aoxue Liu, Boyu Li, Yong Wang & Ziteng Yang

To cite this article: Aoxue Liu, Boyu Li, Yong Wang & Ziteng Yang (2025) Friend Link Prediction Method Based on Heterogeneous Multigraph and Hierarchical Attention, Applied Artificial Intelligence, 39:1, 2427545, DOI: 10.1080/08839514.2024.2427545

To link to this article: https://doi.org/10.1080/08839514.2024.2427545

Published online: 27 Feb 2025.

APPLIED ARTIFICIAL INTELLIGENCE
2025, VOL. 39, NO. 1, e2427545 (37 pages)
https://doi.org/10.1080/08839514.2024.2427545

Taylor & Francis
Taylor & Francis Group

Friend Link Prediction Method Based on Heterogeneous Multigraph and Hierarchical Attention

Aoxue Liu, Boyu Li, Yong Wang, and Ziteng Yang

School of Computer Science, China University of Geosciences, Wuhan, China

ABSTRACT

With the rapid growth of location-based social network (LBSN), rich data comprising social behaviors and location information among users has emerged. Predicting potential friendships accurately from abundant information has become a pivotal research area. While graph neural network (GNN) have shown significant promise in prediction, existing approaches often fail to fully exploit the heterogeneous data characteristics in LBSN. Key challenges include inadequate modeling of the intricate relationships between users and points of interest (POI), overlooking the significance of spatial-temporal information in user trajectories, and underutilizing rich edge features. To address these challenges, we design a novel GRU-enhanced Heterogeneous Multigraph Attention Network (GEHMAN), which is a GNN model enhanced by GRU. We construct a heterogeneous multigraph to comprehensively capture user-POI relationships. We then employ a skip-gram model to embed POI nodes from user sub-trajectories and use RNN with GRU units to embed user nodes. GEHMAN utilize hierarchical attention mechanism to consolidate node information by aggregating diverse types of neighboring nodes and connecting edges. Experiments on six real city datasets show that compared with the best performance of six benchmark methods including LBSN2vec++, Metapath2vec and HAN, the average improvement percentages of GEHMAN in AUC, AP, and Top@K are 2.225%, 1.948%, and 6.353%, respectively.

ARTICLE HISTORY
Received 18 August 2024
Revised 7 October 2024
Accepted 3 November 2024

Introduction

Social networks have developed rapidly in the past few decades and have become an indispensable part of people's lives and work. Social network platforms such as Facebook and Twitter connect billions of people around the world, allowing people to easily share information, exchange ideas, and establish social relationships (Samanta, Dubey, and Sarkar 2021). Location-based social networks (LBSN) have been rapidly popularized and applied in the mobile Internet era due to their unique advantages of combining social relationships and geographic location information. LBSN can not only provide

CONTACT Yong Wang yongwang@cug.edu.cn School of Computer Science, China University of Geosciences, No. 68 Jincheng Street, East Lake New Technology Development Zone, Wuhan, Hubei Province, China

users with personalized location recommendation services but also deeply explore the social relationships between users, providing valuable data support for social network analysis and optimization (Park and Han 2018). Friend link prediction, as an important branch of LBSN research (Cho, Myers, and Leskovec 2011), involves the analysis and prediction of multi-dimensional data such as user behavior, social relationships, and geographic location, which is of great significance for enriching user social relationships and optimizing LBSN services. By accurately predicting potential friend relationships between users, the LBSN platform can not only recommend friends who are more in line with their social needs to users but also further deepen users' social interactions and enhance the user stickiness of the platform (Guo et al. 2018).

Link prediction methods are mainly divided into similarity-based methods, probabilistic methods, machine learning methods, and other methods. Similarity-based methods are the most classic and commonly used methods in the field of link prediction (Wu et al. 2022). Potential links are predicted by calculating the similarity of nodes or edges. Common similarity measurement methods include (Liu et al. 2007): Common Neighbors (CN), Jaccard Coefficient (JC), Adamic/Adar Index (AA), etc. Probabilistic methods predict links based on probabilistic statistical models (Lin et al. 2017), such as the probabilistic tensor decomposition model and the Stochastic Markov model. In recent years, the rapid development of machine learning technology has provided many innovative and powerful methods for solving the friend link prediction problem (Bhatti et al. 2023). Among them, the use of graph neural network (GNN) can effectively capture the complex structural characteristics of nodes and their neighborhoods (Bing et al. 2023), thereby more accurately modeling the intricate social relationships between users. At the same time, deep learning models based on attention mechanism are also widely used in this field. They can adaptively focus on important connections between users, better capture implicit social ties, and show excellent performance in large-scale social network scenarios.

However, although GNN provides an opportunity to predict friend links more accurately and efficiently, how to fuse complex and diverse data from massive and sparse information to effectively extract user features has become a new challenge. Most friend link prediction methods based on GNN do not fully utilize the heterogeneous data of LBSN. For example, Heter-GCN (Wu et al. 2019) and Walk2Friends (Backes et al. 2017) mainly use simple graphs with limited connection types or heterogeneous homogeneous graphs to model the relationships in LBSN, ignoring geographic location and time information. So, they are unable to restore the complexity and diversity of relationships in the real world. MVMN (Zhang, Lai, and Wang 2020) distinguishes social networks from user trajectories, ignoring the importance of spatial-temporal information of user trajectories in predicting friend links. In

addition, many graph models such as GraphSAGE (Hamilton, Ying, and Leskovec 2017) focus on extracting features at the node level, ignoring the edge features that are closely related to the link prediction task.

In order to solve the problems above, we propose a novel GNN model based on heterogeneous multigraph and hierarchical attention and enhance it with GRU, named GRU-enhanced heterogeneous multigraph attention network (GEHMAN). In the process of building this model, we borrow widely recognized methods from previous studies (Fu et al. 2020; Gao et al. 2017; Li et al. 2023; Wang et al. 2019; Wu et al. 2019) and cleverly integrate a variety of technical means, including 1) Taking into account the complexity and diversity of relationships in LBSN, a heterogeneous multigraph with two types of nodes and seven types of edges is constructed. 2) The skip-gram and GRU model are implemented to learn the rich spatiotemporal information contained in the user trajectory to obtain the embedding of users and points of interest (POI), which are served as the input of GNN. 3) The information of neighboring nodes and adjacent edges is fully considered to iteratively update node features, and a multi-head attention mechanism is employed to assign learnable weights in aggregation. 4) The idea of supervised contrastive learning is applied for model training. To verify the effectiveness of the GEHMAN model, experiments are conducted on six real city datasets. Comparative experiments with multiple baseline methods demonstrate that the model effectively improves the performance of friend link prediction.

The remainder of this paper is organized as follows. Section ("Background") introduces the background of our study. Section ("Problem definition") formally defines the problem of friend link prediction in LBSN, which is solved by the proposed GEHMAN as described in section ("Methodology"). In Section ("Experimental analysis"), we detail the extensive experiments performed to verify the performance of GEHMAN on real-world social networks. Section ("Conclusion and future work") concludes this paper and outlines future study directions.

Background

Preliminaries

Heterogeneous Multigraph

A heterogeneous multigraph is a network model that consists of multiple types of nodes and edges, allowing multiple edges of different types to connect two nodes. The edges in this type of graph can be directed or undirected. Additionally, nodes and edges in a heterogeneous multigraph can have attributes. This graph structure can represent the diversity and rich interaction relationships in complex networks.

Hierarchical Attention

Hierarchical attention is an information aggregation method in GNN that aims to improve the performance of node representation. It is divided into two levels. First, edge-level attention aggregates information about the target node and adjacent nodes connected to the same edge type. Different weights are assigned according to the importance of each adjacent node to generate a node representation based on the edge type. Next, semantic-level attention assigns weights to different types of edges and integrates these diverse node representations. Finally, a comprehensive representation of the target node is formed. Through this dual aggregation, the hierarchical attention mechanism can effectively integrate information from different nodes and different edge types, enhancing the semantic understanding ability of the node.

GRU

The Gated Recurrent Unit (GRU) is a recurrent neural network (RNN) architecture for processing sequence data. It aims to solve the vanishing gradient problem faced by traditional RNN in long sequence learning. GRU controls the flow of information and the updating of memory by introducing update gates and reset gates. The update gate determines the degree of influence of the current input and previous state on the current state, while the reset gate controls the influence of the previous state in generating the current state. Due to its relatively simple structure and high computational efficiency, GRU performs well in many sequence modeling tasks.

Approximate Activity Center

Average the locations of the top 30% of POIs in terms of user$u_i$visit times to obtain an approximate activity center for the user u_i. Users with similar approximate activity centers have a greater chance of becoming friends.

Activity Association

The distance threshold is set as δ. If the location of a POI is within a distance of δ from user's approximate activity center, it can be considered that there exists an activity association between useru_i and POI v_j.

Segmentation of the Day

Most cities have a peak check-in time at dinner time, and also show a periodic check-in at other times. In order to better statistically analyze the time characteristics of user check-in behavior, we divide a day into four periods, namely 5:00-10:00 for breakfast time, 10:00-15:00 for lunch time, 15:00-22:00 for dinner time, and 22:00-5:00 for nightlife time.

Related Work

As an important branch of LBSN research, friend link prediction has attracted a lot of attentions. Many useful methods can be applied to solve the problem, and commonly used methods include similarity-based methods and deep learning-based methods.

The similarity-based approach is the most commonly used approach for link prediction which is developed based on the assumption that two nodes in a graph interact if they are similar (Liu et al. 2007). In recent years, the SNGNN framework proposed by Zou et al. (2023) optimizes the neighborhood aggregation process by combining the node similarity matrix, improves the performance of node classification, and performs well on heterogeneous datasets. Similarity-based link prediction methods can be implemented through techniques such as random walks and community detection (Daud et al. 2020). The random walk technique is an effective method for graph analysis, which evaluates the similarity between nodes by simulating multiple random walks to calculate the probability of reaching each other. This approach captures the implicit associations between nodes and facilitates the prediction of new link formations. Several notable algorithms, such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2Vec (Grover and Leskovec 2016), are based on the principles of random walks. DeepWalk learns node embeddings by generating contexts through random walks on the graph, while Node2Vec optimizes the walk strategy to balance the retention of local and global structural information, ultimately resulting in higher-quality node representations. These algorithms have demonstrated exceptional performance in various applications, including social networks and recommendation systems. Recent studies such as Berahmand et al. (2022) proposed an effective method to improve local random walks, which is to encourage random walks to move to more influential nodes at each step to improve the link prediction performance in social networks. Community detection has long been a crucial area of research in network analysis. The objective of community detection is to partition the nodes of a network into distinct communities, indicating potential relationships among nodes. Typically, if two nodes belong to the same community, they are more likely to form new links. One of the earliest community detection algorithms is the edge betweenness-based method proposed by Girvan and Newman (2002). Their work demonstrated that community structures are prevalent in social and biological networks by identifying critical edges in the network. This research laid the foundation for subsequent studies in the field. More recently, Ma and Nandy (2023) used Using Contextual Multilayer Networks for community detection to improve detection performance. Ma et al. (2020) proposed a community detection algorithm based on influential nodes (LGIEM) to

identify the most influential nodes which are considered as cores of communities and achieve the initial communities.

Deep learning methods use neural networks to learn representations of users or links to complete link prediction. The current mainstream deep learning methods are graph representation learning (GRL) and graph neural network (GNN) (Xie et al. 2022). GNN is currently the most popular method to deal with Non-Euclidean Structure Data in deep learning and was first proposed by Scarselli et al. (2008). The main idea of GNN is to iteratively aggregate feature information from neighbors and combine the aggregated information with the current central node representation during the propagation process (Wu et al. 2022). The combined representation is input into a multi-layer neural network to perform multiple downstream tasks such as node classification, link prediction, and graph classification (Li et al. 2024). Previous works in GNN or network embedding mainly focused on the simple graphs (e.g., GCN (Bruna et al. 2013), GAT (Velickovic et al. 2017)), heterogeneous simple graphs (e.g., Metapath2Vec (Dong, Chawla, and Swami 2017), HAN (Wang et al. 2019), HetGNN (Zhang et al. 2019), MAGNN (Fu et al. 2020)) and hypergraph (All Set Transformer (Chien et al. 2021), HGNN (Feng et al. 2019)). It is worth noting that there is now also a modeling method called multigraph. For instance, Wang et al. (2021) forecast ambulance demand with profiled human mobility via heterogeneous multigraph neural networks. Geng et al. (2019) construct a spatial-temporal multigraph convolutional network to predict ride-hailing demand.

In link prediction tasks, GNN combined with attention mechanism to improve performance has been widely used. The landmark starting point for the application of attention mechanism in the field of artificial intelligence is the work of Bahdanau, Cho, and Bengio (2014) in the field of natural language processing. They implemented the attention mechanism in the machine translation model to solve some problems of RNN. Subsequently, the research on attention mechanism in various tasks became popular, such as text classification (Yang et al. 2016), recommendation system (He et al. 2018), etc. The introduction of Transformer model (Vaswani et al. 2017) further promoted the popularization of attention mechanism. It proposed the self-attention mechanism and the multi-head attention mechanism. Although the Transformer model was proposed for machine translation, it was later widely used in other tasks, such as image processing and recommendation system. In addition, attention weights can be calculated not only based on the original input sequence but also based on different levels of abstraction, i.e., hierarchical attention. Wang et al. (2019) proposed a classic hierarchical attention graph neural network model HAN based on heterogeneous graphs. The model uses a double-layer attention that combines node-level attention and semantic-level attention. Node-level attention is used to learn the importance between nodes and neighboring nodes on meta-paths, while semantic-level

attention is used to learn the importance of different meta-paths. The features of adjacent nodes based on meta-paths are then hierarchically aggregated to generate node representations.

Although the development of deep learning has provided many advanced methods for friend link prediction, many studies only use relatively single technical means. They cannot fully explore the complex relationships in LBSN. We cleverly combined multiple technologies to build the GEHMAN model to improve the accuracy of friend link prediction, such as constructing a heterogeneous multigraph to fully capture user–POI relationships, employing skip-gram and GRU models for node initial feature embedding, and referring to the HAN model to use hierarchical attention mechanisms to aggregate different types of neighbor nodes and their connecting edges to integrate node information.

Problem Definition

To facilitate the presentation in this study, Table 1 summarizes the frequently used symbols in this study. A heterogeneous multigraph $\mathcal{G}$is composed of a node set$N$ and an edge set E, denoted as $(N, E, R, C, attrv, attre)$. Here, N represents the node set, E represents the edge set, R and $C$are sets of node types and edge types, and *attrv* and *attre* are attribute functions for nodes and edges, respectively.

In LBSNs data, $U = \{u_1, u_{2j}, \ldots, u_m\}$ represents the set of m users and $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of n POIs. The information of POI $v \in V$ includes its coordinates (latitude and longitude) l_v and the venue category c_v, such as a cinema or restaurant. Each check-in record is represented by (u_i, v_j, t_k), which indicates that the user u_i has accessed the POI v_j at time t_k.

By combining the user's check-in records with the location and category information of the POIs, we can obtain details about the user's individual check-in activities. It is represented as a quintuple $H_{u_i}^{t_k} = \left(u_i, v_j, t_k, l_{v_j}, c_{v_j}\right)$, where the user u_i visited POI v_j at time t_k, with

Table 1. The frequently used symbols in this study.

Notations	Description
$\mathcal{G}$	Heterogeneous multigraph
N, E	Node set and edge set, respectively
R, C	Node types set and edge types set, respectively
$U = \{u_1, u_2, \ldots, u_m\}$ $V = \{v_1, v_2, \ldots, v_n\}$	Set of m users and the set of n POIs, respectively
l_v, c_v, t_k	Coordinate, category and visited time of POI, respectively
$H_{u_i} = \left\{H_{u_i}^{t_1}, H_{u_i}^{t_2}, \ldots, H_{u_i}^{t_k}\right\}$	Historical check-in trajectory of u_i
$\mathcal{H}_U = \{H_{u_1}, H_{u_2}, \ldots, H_{u_m}\}$	Historical check-in trajectories of all m users

the location of v_j being l_{v_j} and its venue category being c_{v_j}. The check-in activities of user $u_i \in U$ form their historical check-in trajectory $H_{u_i} = \left\{H_{u_i}^{t_1}, H_{u_i}^{t_2}, \ldots, H_{u_i}^{t_k}\right\}$, and $\mathcal{H}_U = \{H_{u_1}, H_{u_2}, \ldots, H_{u_m}\}$ represents the historical check-in trajectories of all m users.

Formally, we define the problem as follows: Given a pair of users u_i and u_j, the aim of this study is to leverage the historical check-in records and social relationships of users to construct a heterogeneous multigraph $\mathcal{G}$. On this basis, we aim to learn a model $\mathcal{F}(u_i, u_j) \rightarrow [0, 1]$, where $\mathcal{F}(u_i, u_j)$ represents the probability of establishing a friendship link between u_i and u_j in the future.

Methodology

Most GNN-based friendship link prediction models do not adequately extract and utilize the rich semantic information present in heterogeneous social networks of LBSN. The underlying feature associations related to distance or sequence are not explicitly modeled. Therefore, we propose a novel GRU-enhanced Heterogeneous Multigraph Attention Network (GEHMAN), enhanced by GRU units and combined with a hierarchical attention mechanism.

The overall architecture of the GEHMAN model is shown in Figure 1, which mainly consists of four main components: heterogeneous multigraph construction, node initial feature learning based on skip-gram and GRU, node feature learning based on hierarchical attention, and model training based on the idea of supervised contrastive learning.

First, we extract information from LBSN to construct a heterogeneous multigraph containing two types of nodes (users and POIs) and seven types of edges. Then, the original trajectory of each user is divided into a series of sub-trajectories on a daily basis. A skip-gram model is used to learn POI embeddings. Subsequently, the GRU model is employed to learn user embeddings from user trajectories. In this way, the initial embeddings for POI nodes and user nodes in the heterogeneous multigraph is obtained. Node feature learning employs a hierarchical attention mechanism that integrates both edge-level attention and semantic-level attention to effectively derive the initial node feature representations. Furthermore, multi-head attention is utilized to facilitate the aggregation of these features. After obtaining the final node representations for user nodes, cosine similarity is employed to calculate the likelihood of a potential friend link between these users. Subsequently, positive and negative samples are divided, and the principles of supervised contrastive learning are used to train the model.

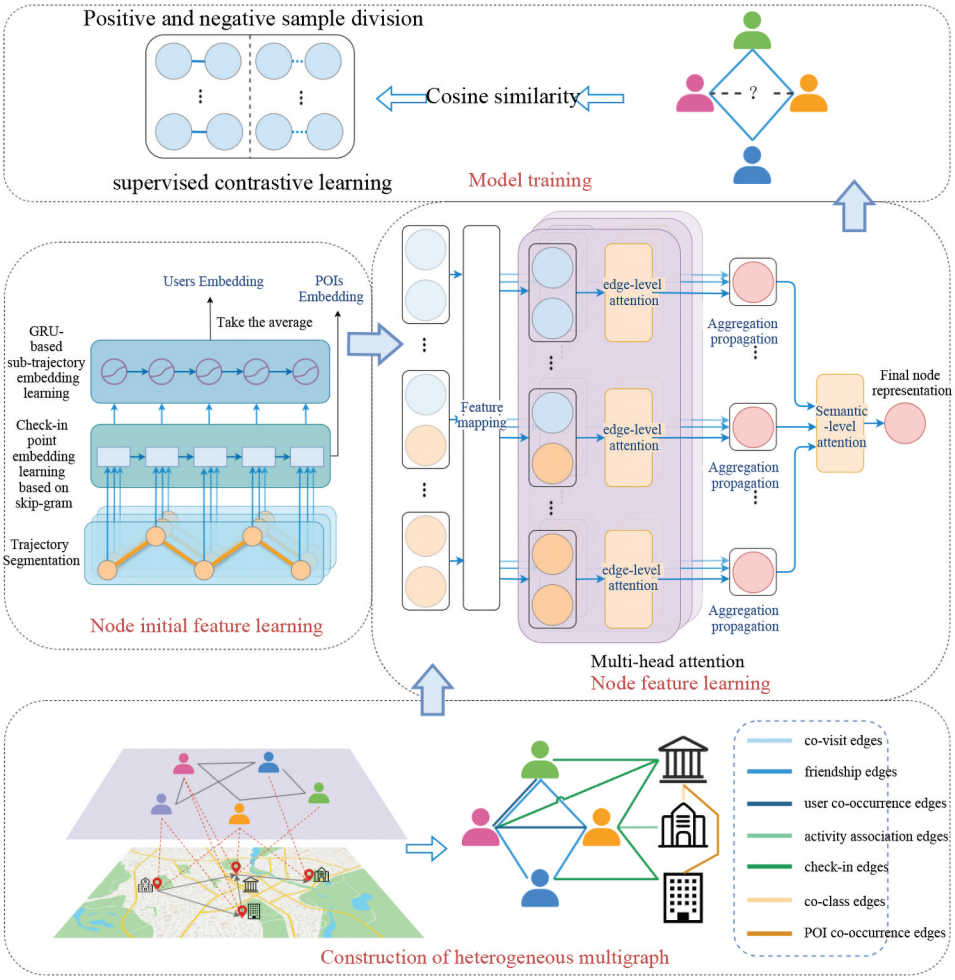

Figure 1. GEHMAN model architecture, consists of four parts: heterogeneous multigraph construction, node initial feature learning based on skip-gram and GRU, node feature learning based on hierarchical attention mechanism, and model training based on supervised contrastive learning idea.

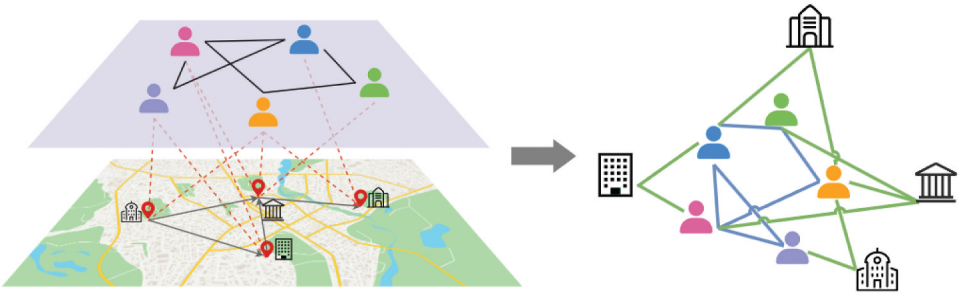

Figure 2. LBSN diagram: the left side shows a complex network of multiple users and geographic locations, with lines representing interactions or relationships between them. The right side simplifies the network to show a clearer structure.

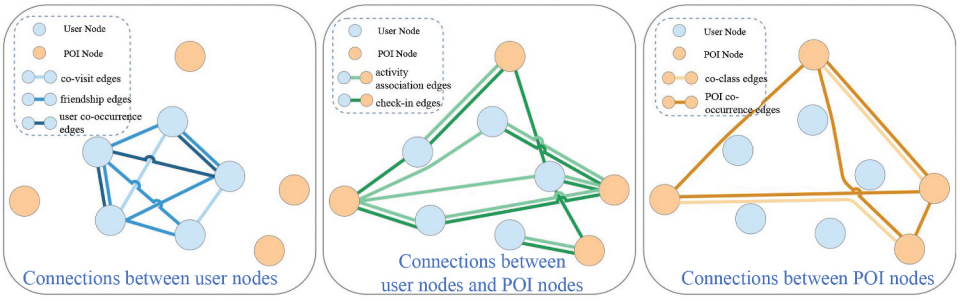

Figure 3. Node relationship diagram: among user nodes, there are friendship edges, co-visit edges, and user co-occurrence edges. Between user nodes and POI nodes, there are activity association edges and check-in edges that highlight user interactions with POIs. Among POIs nodes, there are co-class edges and POI co-occurrence edges, showcasing the relationships between different points of interest.

Construction of Heterogeneous Multigraph

First, the LBSN data is converted into graph data, as shown in Figure 2. Then a user-POI heterogeneous multigraph is constructed, which contains two types of nodes and seven types of edges. The relationships between the nodes are illustrated in Figure 3. Among user nodes, there are friendship edges, co-visit edges, and user co-occurrence edges. Between user nodes and POI nodes, there are activity association edges and check-in edges that highlight user interactions with POIs. Among POI nodes, there are co-class edges and POI co-occurrence edges, showcasing the relationships between different points of interest. Taking one of the user nodes and its neighbor nodes as a sample, the local display structure of the heterogeneous multigraph is shown in Figure 4.

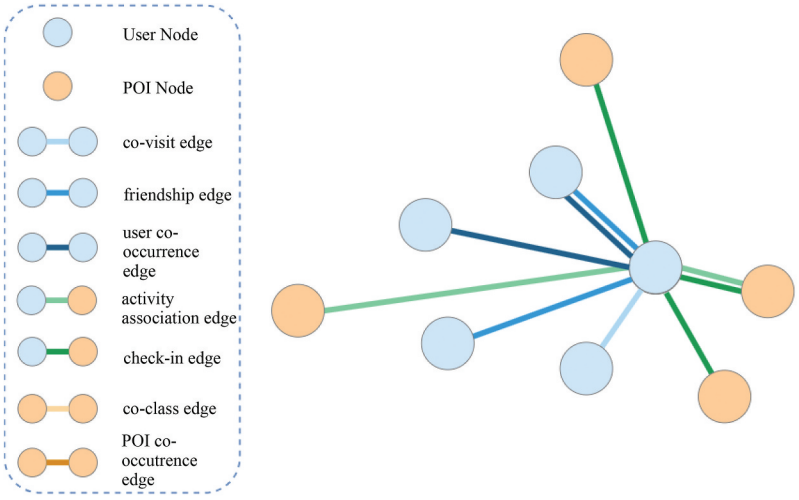

Figure 4. Local diagram of the heterogeneous multigraph, taking a user node as an example.

The definitions of the seven relationship edges and the settings of their initial features are as follows:

Friendship Edge

A friendship edge connects two users who have a friend relationship. Its characteristic number is 1, which is the distance between the approximate activity centers of the two users.

Check-In Edge

A check-in edge connects a user and the POI visited by the user. Its feature number is 5, that is, the frequency of the user's check-in at the POI in the four time periods of breakfast time, lunch time, dinner time, and nightlife time, as well as the distance between the user's approximate activity center and the POI.

Co-Class Edge

A co-class edge connects two POIs with the same venue category. Since each POI in the Foursquare datasets only has one category label, the category information is not comprehensive enough. In our study, we will obtain three categories and three levels of labels for each POI from the Foursquare website, and then reassign the newly obtained category labels to the POIs. That is, each POI can have up to three categories at the same time, and each category contains a three-level classification structure, so that each POI will have up to nine category labels. The number of features of the co-class edge is 10, that is, if there is a corresponding category label among the nine category labels of the two POIs, the feature is set to 1, otherwise to 0. The last feature is set to the distance between the two POIs;

User Co-Occurrence Edge

A user co-occurrence edge connects two users who visited the same POI in the same time interval (setting as 1 hour in our experiment). The feature number of the user co-occurrence edge is 5, which is the number of times the two users co-occur in the four time periods and the distance between the approximate activity centers of the two users.

POI Co-Occurrence Edge

A POI co-occurrence edge connects two POIs that appear in the same user trajectory. The purpose of POI co-occurrence edge is to mine the association between POIs. If a user often visits these two POIs, it means that these two POIs may be closely connected in some way. The number of POI co-occurrence edge features is 2, that is, the number of times these two POIs

appear together in all user trajectories, and the distance between these two POIs.

Co-Visit Edge

A co-visit edge connects two users who have visited the same POI. The feature number of a co-visit edge is 2, i.e., the frequency of the two users' co-visit behavior and the distance between the two users' approximate activity centers.

Activity Association Edge

An activity association edge connects a user to the POI whose distance from the user's approximate activity center is less than the distance threshold δ. Studies have shown (Sun et al. 2021) that 20% of user check-in occur within 1 km, and 60% of user check-in occur between one and ten kilometers. We use activity association edges to capture the spatial association between user activity centers and POIs. Users are more likely to visit the POIs near activity centers. The feature number of an activity association edge is 1, which is the distance between the user and the POI.

In contrast to the HMGCL model (Li et al. 2023), the co-visit edges and the user co-occurrence edges are added between user nodes. Because studies have shown (Cho, Myers, and Leskovec 2011) that approximately 30% of new friendships occur among users who have visited the same location. The relationships between users can be more comprehensively modeled in order to fully exploit potential friendships.

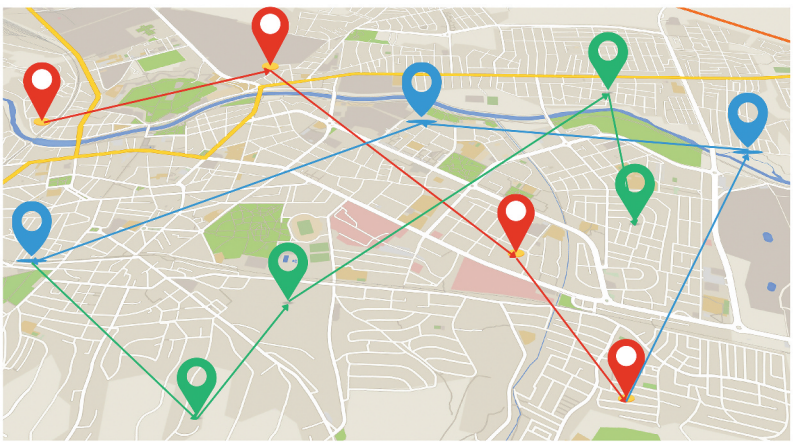

Figure 5. A user trajectory division diagram: the check-in points and trajectory lines of different colors represent different sub-trajectories.

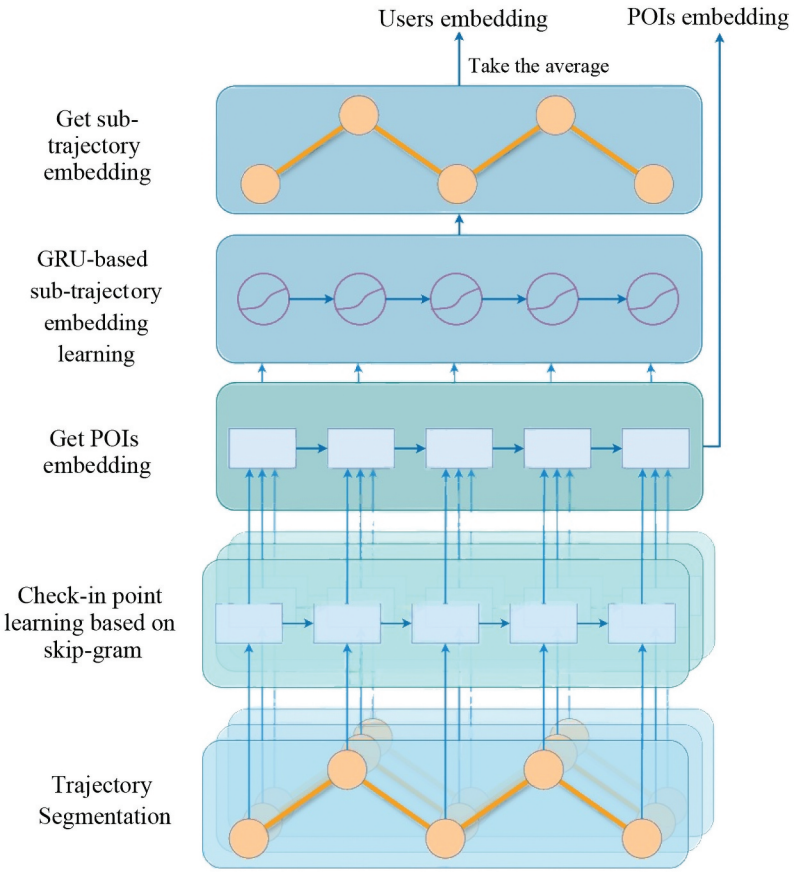

Figure 6. The framework of node initial feature learning: sub-trajectory division, POI embedding using skip-gram model from user sub-trajectories and user embedding with GRU.

Node Initial Feature Learning

In order to provide meaningful node feature inputs to GNN models and introduce temporal information, this section adopts the method from Fu et al. (2020) to learn the embeddings of users and POIs. First, the original trajectory of each user is divided into a series of sub-trajectories on a daily basis. This division can effectively capture the user's daily behavior patterns, reduce data noise, maintain behavior continuity, and improve model training efficiency. And Figure 5 provides an illustration of the concept of user trajectory division. Then, a skip-gram model is used to learn POI embeddings. Subsequently, the GRU model is employed to learn user embeddings from user trajectories. In this way, the initial embeddings of POI nodes and user nodes in the heterogeneous multigraph is obtained. The module structure is illustrated in Figure 6.

We use GRU as a model component. Compared with the long short-term memory network (LSTM), GRU has similar performance but fewer

parameters and higher computational efficiency. This can better adapt to the characteristics of our data and the needs of practical applications.

POIs Embedding

We represent the historical check-in records $H_{u_i} = \left\{H_{u_i}^{t_1}, H_{u_i}^{t_2}, \ldots\right\}$ of each user $u_i \in U$ in the LBSN data as a user trajectory sequence T_{u_i}. To reduce the computational complexity and capture richer semantics of moving patterns, the original trajectory T_{u_i} of each user u_i is segmented by day to obtain a series of continuous sub-trajectories $T_{u_i} = \left\{T_{u_i}^1, T_{u_i}^2, \ldots\right\}$.

Afterward, we utilize the skip-gram model to learn embeddings for POIs based on the sub-trajectories of all users. The skip-gram model is a model for learning word embeddings. It is able to map words into a low-dimensional vector space while maintaining the semantic and grammatical relationships between words. We regard POIs as words and user sub-trajectories as sentences. The length of the sub-trajectory is derived from the user check-in data divided by day, and thus it is not fixed. Then use the skip-gram model to learn POIs embeddings.

To introduce more temporal information, the initial feature Ψ_{v_j} of POI v_j includes the geographical location l_{v_j}, the venue category c_{v_j}, and the average of check-in time difference Δt between the check-in time of POI v_j and the previous one of POI v_{j-1} in different sub-trajectories.

For a given POI v_j, its context is defined as $C(v_j)$, where $C(v_j) = \{v_j - c, \ldots, v_j + c\}$, and c is the size of the sliding window. In the given context $C(v_j)$, the embedding Ψ'_{v_j} of POI$v_j$is learned by maximizing the probability of predicting its context, that is, the probability that the context POI co-occurrence with v_j in the sub-trajectory $T_{u_i}^k$ where POI v_j is located. The formula is as follows.

$$max \prod_{v_l \varepsilon C(v_j)} \frac{exp\left\{\Psi_{v_j}^T \Psi'_{v_l}\right\}}{\sum_{v=1}^{|c|} exp\left\{\Psi_{v_j}^T \Psi'_{v}\right\}} \tag{1}$$

For each POI v_j, the average of the learned embedding vectors Ψ'_{v_j} containing v_j in all sub-trajectories is computed as the final embedding vector $\Psi_{v_j}^{final}$ for POI v_j.

Users Embedding

After learning all the POIs embeddings, the user's sub-trajectory $T_{u_i}^k \in T_{u_i}$, which obtains the POIs embeddings, is input into RNN with GRU units. The sub-trajectory $T_{u_i}^k$ consists of k time steps, and $x_{k,t}$ represents the feature input of sub-trajectory $T_{u_i}^k$ at time step t, that is, the embedding of POI v denoted as

$\Psi_v^{final} \in \mathbb{R}$, where $\mathbb{R}$ represents the set of real numbers. The hidden state $h_{k,0} \in \mathbb{R}$ is initialized first, and then for each time step t, the reset gate $r_{k,t}$, the update gate $z_{k,t}$, the candidate hidden state $\tilde{h}_{k,t}$ are computed to update the hidden state $h_{k,t}$ by

$$r_{k,t} = \sigma\left(W_r \cdot \left[h_{k,t-1}, x_{k,t}\right] + b_r\right) \tag{2}$$

$$z_{k,t} = \sigma\left(W_z \cdot \left[h_{k,t-1}, x_{k,t}\right] + b_z\right) \tag{3}$$

$$\tilde{h}_{k,t} = tanh\left(W_h \cdot \left[r_{k,t} \cdot h_{k,t-1}, x_{k,t}\right] + b_h\right) \tag{4}$$

$$h_{k,t} = \left(1 - z_{k,t}\right) \cdot h_{k,t-1} + z_{k,t} \cdot \tilde{h}_{k,t} \tag{5}$$

where $W_r \in \mathbb{R}^{\times}$, $W_z \in \mathbb{R}^{\times}$ and $W_h \in \mathbb{R}^{\times}$ are the weight matrices for the reset gate, update gate, and candidate hidden state, respectively. $b_r \in \mathbb{R}$, $b_z \in \mathbb{R}$ and $b_h \in \mathbb{R}$ are the bias vectors for the reset gate, update gate, and candidate hidden state, respectively. $tanh(\cdot)$ is the hyperbolic tangent function, and $\sigma(\cdot)$ is the nonlinear action function of neurons, the sigmoid function.

The embedding of the final sub-trajectory $T_{u_i}^k$ is represented by the hidden state at the last time step h_{k,T_k}. The embedding E_{u_i} of user u_i is obtained as the average of embeddings of all sub-trajectories of that user u_i as follows.

$$E_{u_i} = \frac{1}{N} \sum_{N}^{k=1} h_k \tag{6}$$

E_{u_i} is the embedding of user u_i. N is the total number of sub-trajectories of user u_i. h_k is the final hidden state of the k-th sub-trajectory of user u_i.

Node Feature Learning

The node feature learning section consists of feature mapping, edge-level attention, aggregation propagation, and semantic-level attention, as shown in Figure 7.

Feature Mapping

The initial node features and initial edge features in the user-POI heterogeneous multigraph may not be uniform in terms of scale range, dimension, and feature space. It is necessary to perform type-specific linear transformations on nodes and edges to map their features to a unified dimension and feature space. The formulas for feature mapping are as follows.

$$h_v = W_{r_v} \cdot X_v \tag{7}$$

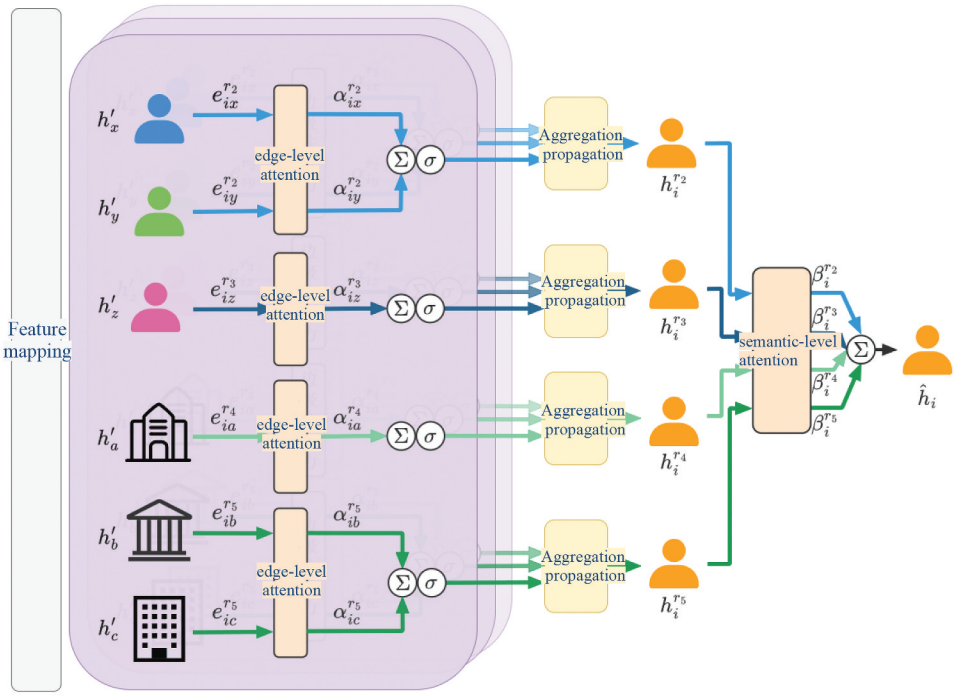

Figure 7. Node feature learning uses a hierarchical attention mechanism consisting of edge-level attention and semantic-level attention to learn the initialized node feature representation.

$$h_e = W_{c_e} \cdot X_e \tag{8}$$

where $h_v \in \mathbb{R}^{d_{note}}$ is the mapped feature of node $v \in V$, $X_v \in \mathbb{R}^{r_v}$ is the original feature of node v, $r_v \in R$ is the node type of node v, $W_{r_v} \in \mathbb{R}^{d_{note} \times d_{\cdot v}}$ is the mapping matrix specific to node type r_v. $h_e \in \mathbb{R}^{d_{edge}}$ is the mapped feature of edge $e \in$ E, $X_e \in \mathbb{R}^{c_e}$ is the original feature of edge e, $c_e \in C$ is the edge type of edge e, $W_{c_e} \in \mathbb{R}^{d_{edge} \times d_{ce}}$ is the mapping matrix specific to edge type c_e.

Edge-Level Attention

Since the information conveyed by different neighboring nodes through an edge of a certain type is not of the same importance to the target node, edge-level attention is used to learn the attention weights of the features of the neighboring nodes connected by edges of a certain type.

Assuming we have a target node $i \in V$, which is connected to neighboring node through multiple edges of different types. Taking an example of one neighbor node $j \in V$ and one edge $e_{ij}^{c_k}$ connecting target node i to neighbor node j, where $c_k \in C$ represents the edge type. Next, we calculate the edge-level attention weight $\alpha_{ij}^{c_k}$, which represents the importance of neighbor node$j$to the

target node i through edge $e_{ij}^{c_k}$ of type c_k. The calculation formulas are as follows.

$$\eta_{i,j}^{c_k} = \mathfrak{a}_{c_k}^{T} LeakyReLU\left(W_E^{c_k}\left[h_i \parallel h_j \parallel h_{e_{i,j}^{c_k}}\right]\right) \tag{9}$$

$$\alpha_{i,j}^{c_k} = \frac{\exp\left(\eta_{i,j}^{c_k}\right)}{\sum_{n\in N_i^{c_k}} \exp\left(\eta_{i,j}^{c_k}\right)} \tag{10}$$

where $\alpha_{i,j}^{c_k}$ is the edge-level attention weight, $N_i^{c_k} \in V$ is the set of neighbor nodes connected to the target node i through edges of type c_k, $h_i, h_j, h_n \in \mathbb{R}^{d_{note}}$ are the features of nodes i, j, n respectively, $h_{e_{i,j}^{c_k}} \in \mathbb{R}^{d_{edge}}$ is the feature of edge $e_{i,j}^{c_k}$, $h_{e_{i,n}^{c_k}}$ is the feature of edge $e_{i,n}^{c_k}$ connecting nodes i and n with edge type c_k, $\mathfrak{a}_{c_k}^{\mathfrak{T}} \in \mathbb{R}^{d}$ is the learnable edge-level attention vector specific to edge type c_k, $W_E^{c_k} \in \mathbb{R}^{d\times(\nvDash d_{note} d_{\eth})}$ is the learnable edge-level attention weight matrix specific to edge type c_k, $\parallel$ is concatenation operation, $LeakyReLU(\cdot)$ is the activation function.

Aggregation Propagation

After obtaining the edge-level attention weights, the neighboring node features need to be aggregated. Traditional GNN only fuse node features in aggregation. In this section, the aggregation of edge features is added on the basis of traditional GNN aggregation, i.e., the information of both neighboring nodes and connected edges is fused. Meanwhile, in order to better understand the complex structure and relationship of heterogeneous multigraph and to enhance the expressive ability of the model, the neighbor aggregation of the layer will be completed by using multi-head attention. The formula is as follows.

$$\hat{h}_i^{c_k(l)} = \|_{m=1}^{M} \sigma\left(W_A^{c_k(l)} \sum_{n\in N_i^{c_k}} \alpha_{i,n}^{c_k(l)} \cdot \left[h_n^{(l)} \parallel h_{e_{i,n}^{c_k}}^{(l)}\right]\right) \tag{11}$$

where l is the l-th propagation layer, $\hat{h}_i^{c_k(l)}$ is the neighboring aggregation feature of the layer, $M$is the number of attention heads, $W_A^{c_k(l)}$ is the learnable multi-head attention weight matrix of the layer, $N_i^{c_k} \in V$ is the set of neighboring nodes connected to the target node i through the edges of type c_k, $\alpha_{i,n}^{c_k(l)}$ is the edge-level attention weight of the neighboring node n of this layer to the target node i based on the c_k type edge, $h_n^{(l)}$ is the feature of the neighboring node$n$of the layer, $h_{e_{i,n}^{c_k}}^{(l)}$ is the feature of the connecting edge $e_{i,n}^{c_k}$ of the layer, and $tanh(\cdot)$ is selected as the activation function.

After obtaining the neighboring aggregation feature, it needs to be passed to the target node to obtain a kind of node semantic representation $h_i^{c_k} \in \mathbb{R}^{\mathrm{d}}$ of the target node i of the layer based on the edge type c_k. The formula is as follows.

$$h_i^{c_k(l)} = ReLU\left(W_p^{c_k(l)}\left[h_i^{c_k(l-1)} \parallel \hat{h}_i^{c_k(l)}\right]\right) \tag{12}$$

where $W_p^{c_k(l)}$ denotes the learnable transformation matrix of the layer and $h_i^{c_k(l-1)}$ is the node semantic representation of the target node i of the previous layer based on the edge type c_k.

The output of the last layer L of the propagation layer is used as the node semantic representation of the target node i based on the edge type c_k, denoted as $h_i^{c_k(L)}$.

Semantic-Level Attention

After computing all the edge-level attention and aggregation propagation for a target node i with n types of edge connections, the semantic representations of n target nodes based on different types of edges will be obtained. In order to learn a more comprehensive node representation, this study will use semantic-level attention to aggregate these n target nodes semantic representations based on specific types of edges, learn their respective weights, and compute their weighted sums to obtain the final target nodes representation, where the semantic-level attention is inspired by HAN (Wang et al. 2019). However, HAN automatically learns the importance of different meta-paths and fuse information for the specific task. It focuses on global semantic attention and is based on the idea of meta-path, which has certain limitations. In the user-POI heterogeneous multigraph constructed in this study, each user has different topological structures and neighborhood semantics. The GEHMAN model does not rely on meta-paths, and semantic-level attention focuses more on local structures.

The importance of the semantic representation of the target node based on the edges of type c_k to the target node i, i.e., the semantic-level attention weight, can be denoted as $\beta_i^{c_k}$. The calculation formulas are as follows.

$$\omega_i^{c_k} = q_S^T \cdot tanh\left(W_S \cdot h_i^{c_k(L)} + b_S\right) \tag{13}$$

$$\beta_i^{c_k} = \frac{exp\left(\omega_i^{c_k}\right)}{\sum_{y\varepsilon n} exp\left(\omega_i^{c_y}\right)} \tag{14}$$

Where L represents the last layer of the propagation layer. $h_i^{c_k(L)}$ as a whole, represents the node semantic representation of the target node i based on the edge type c_k, and is also the output of the last layer L of the propagation layer, which are mentioned below Equation 12. q_S^T is the learnable semantic-level

attention vector, W_S is the learnable semantic-level weight matrix, b_S is the learnable semantic-level bias vector, n is the n semantic representations of the target node i, and $h_v^{c_x}$ is the features of the node v based on the edges of type c_x.

After obtaining the weights of the semantic representation of each target node, the weighted sum is calculated to obtain the final node representation of the target node i by

$$\hat{h}_i = \sum_{y \epsilon n} \beta_i^{c_y} \cdot h_i^{c_y} \tag{15}$$

Model Training

After obtaining the final node representations of user node i and user node j, the cosine similarity is used to calculate the likelihood $s(i,j)$ that there is a potential friend link between user i and user j. The formula is as follows.

$$s(i,j) = \frac{\hat{h}_i \cdot \hat{h}_j}{\| \hat{h}_i \| \cdot \| \hat{h}_j \|} \tag{16}$$

where $\hat{h}_i$, $\hat{h}_j$ are the final features of user i and user j respectively, and $\| \cdot \|$ is the vector modulus.

Self-supervised learning can learn better features, but it still has shortcomings. It does not take into account the feature correlation between users with friend links. In order to make the features of the two users with friend links more similar and the features of the two users without friend links more different, this study will use the idea of supervised contrastive learning to train the model. The original social network datasets are divided into positive and negative samples as the training set, and the positive samples are the pairs of users who have friend links, and the pairs of users who do not have friend links as the negative samples. However, we found that if the negative samples are randomly selected from the pairs of users who do not have friend links in LBSN with sparse friend relationships, the probability of these pairs becoming friends with each other is low and will be easily distinguished by most methods, which reduces the recommendation value in real applications. Moreover, since new friendships are mostly generated in the closer social circle, i.e., between friends of friends, it is more realistic to predict whether second-order or third-order friends will form friendships with each other. Based on the above considerations, the model is trained by randomly selecting negative samples from pairs of users in the social network who are close but not friends, and the Adam algorithm is used for model optimization.

Although the cross-entropy loss function has been widely used in classification tasks, it has been shown that the cross-entropy loss function can lead to poor generalization performance and lack of robustness to noisy labels or

adversarial samples (Chien et al. 2021). Therefore, a joint loss function $\mathcal{L}$ will be used in this study as follows, which contains three loss functions such as Contrastive Loss $\mathcal{L}_{con}$, Binary Cross Entropy Loss $\mathcal{L}_{bce}$, and Margin Loss$\mathcal{L}_{mar}$, which are represented in the following equations.

$$\mathcal{L}_{con} = -\frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}}\left(log\frac{\sum_{(i,j)\in\mathbb{P}} exp(s(i,j)/t)}{\sum_{(i,k)\in\{\mathbb{P}\cup\mathbb{N}\}} exp(s(i,k)/t)}\right) \tag{17}$$

$$\mathcal{L}_{bce} = -\frac{1}{|\mathcal{V}|}\sum_{(i,j)\in\{\mathbb{P}\cup\mathbb{N}\}}\left(y_{i,j}\cdot log(s(i,j)) + \left(\left(1-y_{i,j}\right)\cdot log(1-s(i,j))\right)\right) \tag{18}$$

$$\mathcal{L}_{mar} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}}\sum_{(i,j)\in\mathbb{P}}\sum_{(i,k)\in\mathbb{N}} max(0, 1-s(i,j)+s(i,k)) \tag{19}$$

$$\mathcal{L} = \lambda_1\cdot\mathcal{L}_{con} + \lambda_2\cdot\mathcal{L}_{bce} + \lambda_3\cdot\mathcal{L}_{mar} \tag{20}$$

where $s(i,j)$ is the cosine similarity between user i and user j, i.e., the possibility of user i and user j becoming friends, V is the set of user nodes, and t is the temperature parameter, which is used to control the smoothing degree of the output distribution of the SoftMax function. $y_{i,j} \in \{0,1\}$ denotes the positive and negative sample labels between user i and user j. If it is a positive sample, then $y_{i,j} = 1$, and if it is a negative sample, then $y_{i,j} = 0$. $\mathcal{L}$ is the joint loss function, and $\lambda_1, \lambda_2, \lambda_3$ are the weights of the three losses, respectively.

The contrastive loss and margin loss can make the distance between positive samples become smaller and the distance between negative samples larger. If this distance is less than a certain threshold, the distance is made greater than the threshold by mutual exclusion. This joint loss function can improve the defects of cross-entropy loss and is more suitable for supervised contrast learning of the model in our study.

Experimental Analysis

Experimental Setup

Datasets

The datasets used in this experiment are from Yang et al. (2019, 2020) collected from the real-world Foursquare,[1] which is currently the dominant location-based social network datasets. The datasets collect 22,809,624 check-ins at 3,820,891 POIs from 114,324 users globally for a total duration of 22 months, starting from April 2012 and ending in January 2014, as well as

Table 2. Heterogeneous multigraph data statistics.

Datasets	NYC	TKY	SP	JK	IST	KL
Number of nodes	7,380	18,022	10,066	14,989	22,601	17,128
Number of edges	2,018,590	6,203,675	2,260,581	5,394,713	9,853,857	5,114,992

snapshots of users' social networks before and after the collection of the two time points. Taking into account regional and cultural differences, the experiment selected six city datasets that contain a large amount of social relations and check-in information in the original dataset. They are New York City (NYC), Tokyo (TKY), Sao Paulo (SP), Jakarta (JK), Istanbul (IST), and Kuala Lumpur (KL).

Before dividing the dataset into training set, validation set and test set, the datasets of these six cities need to be preprocessed to remove redundant information and improve the quality of experimental data. Low-frequency POIs with less than 10 check-in records are removed, inactive users with less than 10 check-in records are removed, and more detailed classification of POIs is obtained from the Foursquare website. Table 2 shows the relevant data statistics of the six city datasets after preprocessing. The heterogeneous multigraph has a total of 90,186 nodes and 30,846,408 edges. After that, the positive and negative samples are divided for supervised comparison learning. All user pairs with friend relationships are taken as positive samples, and the same number of samples is randomly selected from second-order or third-order friend user pairs as the negative sample datasets. Finally, the datasets are randomly partitioned into training set, validation set, and test set with the ratio of 80%, 10%, and 10%, respectively.

Comparison Method

In order to demonstrate the effectiveness of the methods, six typical models selected from two classes of methods, graph representation models, and GNN models are used as benchmark models for comparison with GEHMAN. The first class of graph representation learning models include Walk2friends (Backes et al. 2017), Metapath2vec (Dong, Chawla, and Swami 2017) and LBSN2vec++ (Yang et al. 2020), which are mostly used to learn node representations by using different random walk strategies and skip-gram models. The second class of representation learning models based on GNN include Heter-GCN (Yu, Wang, and Li 2018), HAN (Wang et al. 2019) and MAGNN (Fu et al. 2020), which are methods that use the excellent performance of GNN in graph data learning to learn more accurate node representations.

The following are the specific reproduction methods of the six models, and the parameters are consistent with the original papers.

Walk2friends: A user-location bipartite graph is constructed, where users and locations are nodes and interactions are represented by edges. Then, a random walk algorithm is used to generate multiple random walk sequences

starting from the user nodes. These walk sequences are input into the Word2Vec model as training data to train user embeddings in a skip-gram manner. Finally, the learned user embeddings are applied to the friend link prediction task.

Metapath2vec: A classic graph representation learning method based on heterogeneous simple graphs. It first defines the required meta-paths, then generates random walk sequences based on the meta-paths, and finally inputs the obtained sequences into a heterogeneous skip-gram model to learn node representations. We construct a heterogeneous simple graph containing only three types of edges: friendship edges, check-in edges, and POI co-occurrence edges. Then four meta-paths, "user-user," "user-POI-user," "POI-user-POI," and "POI-POI," are selected to learn user embeddings. Finally, the cosine similarity between user node embeddings is calculated for friend prediction.

LBSN2vec++: The LBSN data is modeled as a heterogeneous hypergraph, including homogeneous edges between users (indicating friendship relationships) and heterogeneous hyperedges of user-time-POI-semantics (indicating user check-ins). Then, the "random walk with stay" strategy is used to sample the sequence of user check-ins and social relationships on the hypergraph. The embedded representation of the nodes is learned based on the obtained sequence. Finally, the friend relationship is predicted based on the cosine distance between the node representations.

Heter-GCN: We construct a user mobility heterogeneous graph. It can be divided into user layer, location layer, and user–location interaction layer. It extracts node features from trajectories to provide semantic information for subsequent embedding learning. Then, Graph convolutional networks (GCNs) are used to integrate node features and graph structure information to learn user embedding. This process includes unsupervised learning and semi-supervised learning.

HAN: A classic layered attention graph neural network model based on heterogeneous graphs. As with Metapath2vec, we construct a heterogeneous simple graph containing only three types of edges and four meta-paths. Then, the features of neighboring nodes based on meta-paths are hierarchically aggregated to generate node representations.

MAGNN: A graph neural network based on meta-path aggregation for heterogeneous graph embedding learning. It includes three main parts, namely node content conversion, meta-path internal aggregation, and meta-path inter-aggregation. Thus, information such as node attributes, meta-path internal nodes, and multiple meta-paths are included in the model learning scope. A heterogeneous simple graph built using Metapath2vec and defining the same four meta-paths.

Table 3. GEHMAN model parameter setting.

Parameter name	Parameter Description	Parameter Value
Window_size	Skip-gram window size	10
Initial_embedding_dim	Initial embedding dimensions	64
GRU_num_layers	Number of GRU layers	1
Node_embedding_dim	Node feature map dimensions	256
Edge_embedding_dim	Edge feature map dimensions	10
UserCo_time_thr	User co-occurrence time threshold	1h
ActAssoc_dist_thr	Activity association distance threshold	2km
$\sigma(\cdot)$	Non-linear activation function	$ReLU(\cdot)$
Learn	Adam optimizer learning rate	0.00001
Epochs	Iterations	3000
K	Number of multi attention heads	6
GNN_num_layers	Number of GNN layers	2
t	Contrast loss temperature parameter	0.2
$\lambda_1, \lambda_2, \lambda_3$	Weight parameters in the joint loss function	1,2,3

Parameter Settings

The main parameters involved in the GEHMAN model during the experiment are shown in Table 3. The impact of some parameter values on the model effect will be discussed in the section of related experiments. It is worth mentioning that we use the deep learning framework PyTorch to train the model.

In addition, the parameters in benchmark models are kept as consistent as possible with those in the original paper in order to achieve the best effect.

Evaluation Metrics

The purpose of this paper is to predict the probability that two users will become friends, so this paper uses three evaluation metrics that are widely used in recommend systems to assess model effectiveness.

AUC (Area Under the Receiver Operating Characteristic Curve). AUC is used to measure the ability of the classification model to distinguish between positive and negative samples, and its value is between 0 and 1, with larger values indicating better model performance.

The advantage of AUC is that it avoids the need for classification thresholds and converts predicted probabilities into categories. Moreover, it does not focus on the specific score, but only on the prediction ranking results, so it is more suitable for the model effectiveness evaluation in this paper, and its calculation formula is as follows:

$$AUC = \frac{\sum_i^{M+N}(s_p > s_n) + 0.5 * \sum_i^{M+N}(s_p \leq s_n)}{M * N} \tag{21}$$

where M and N are the number of positive samples and the number of negative samples, respectively, s_p and s_n are the positive sample score and the negative sample score, respectively. $(s_p > s_n)$ and $(s_p \leq s_n)$ denote a set of

positive-negative sample pairs in which the positive sample score is greater than the negative sample score or less than or equal to the negative sample score, respectively.

AP (Average Precision). AP represents the quality of recommendation sorting, and its value ranges from 0 to 1. The higher the value of AP, the better the quality of recommendation sorting. The advantage of AP is that it takes into account not only the accuracy of the recommendations but also the differences in the order of the recommendations. Its calculation formula is as follows:

$$AP = \frac{1}{N}\sum_{k=1}^{N}(P(k) \cdot rel(k)) \tag{22}$$

Where N is the total number of relevant recommendations, $P(k)$ is the precision of the first k recommendations, and $rel(k)$is the relevance of the k-th recommendation, where $rel(k) = 1$ if relevant, and $rel(k) = 0$ otherwise.

Top@k Accuracy Score. Top@K accuracy measures the proportion of correct predictions among the top K results predicted by the model. Its value ranges from 0 to 1. Higher values indicate better performance of top recommendations. The advantage of the Top@K accuracy metric is that it is more realistic. Because in practical applications, users may only pay attention to the first few items in the recommendation list. Therefore, the model uses the Top@K accuracy to evaluate the performance of the model's first K recommendations, and the formula is as follows:

$$Top@K = \frac{m}{K} \tag{23}$$

where $K$is the total number of results for which the first K recommendations are selected and m is the number of correct recommendations among the first K recommendations.

Performance Comparison

Comparison Experiment

Quantitative Analysis. In order to evaluate the effectiveness of the GEHMAN models, Walk2friend, Metapath2vec, LBSN2Vec++, Heter-GCN, HAN, and MAGNN are selected as the baseline methods to be compared in terms of the three evaluations metrics such as AUC, AP, and Top@K. In this section, 10 repetitions of experiments are conducted for each model and the average of the AUC, AP, and Top@K results obtained from the 10 repetitions will be taken as the final results, which are shown in Figures 8, Figures 9 and 10, respectively.

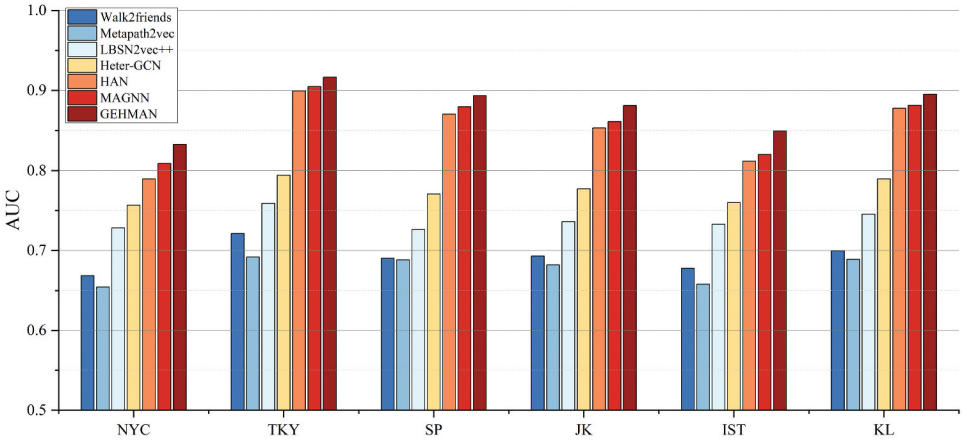

Figure 8. Comparative analysis of AUC across seven recommendation models on six datasets.

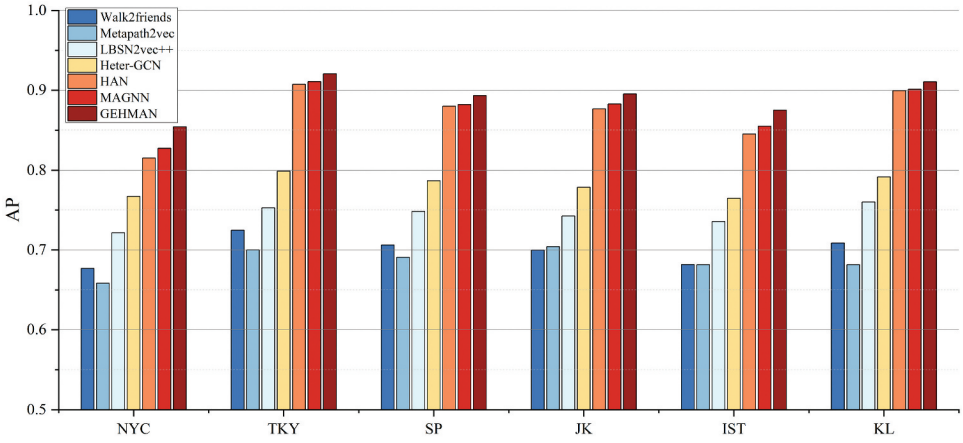

Figure 9. Comparative analysis of AP across seven recommendation models on six datasets.

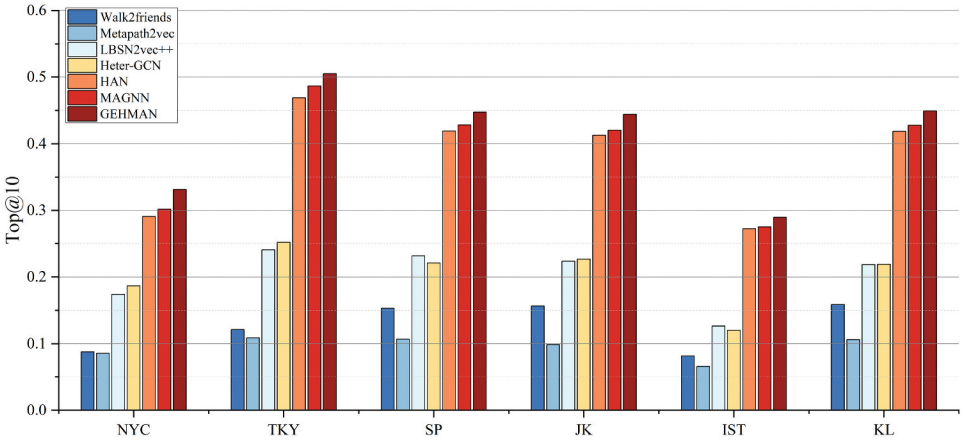

Figure 10. Comparative analysis of Top@10 across seven recommendation models on six datasets.

As can be seen from the figure, the GEHMAN model performs better than the other six comparison models on the datasets. Compared with the best performance among the comparison models, the average improvement percentage of the GEHMAN model in the three indicators of AUC, AP, and Top@K of the six datasets are 2.225%, 1.948%, and 6.353%, respectively. Compared with other datasets, the improvement of GEHMAN model on NYC datasets and IST datasets are relatively more obvious. This may be due to the fact that the NYC datasets and the IST datasets have the least number of users, POIs, check-ins, etc. in one and the most in the other. In this case, the GEHMAN model we proposed can more fully mine the spatial-temporal data features and semantic features in LBSN data because it constructs a heterogeneous multigraph based on user trajectories.

In the comparison experiments of the seven models, the worst performance is the Metapath2vec model. This is because Metapath2vec is an unsupervised learning method that learns the relationship between nodes only through a heterogeneous skip-gram model. It does not distinguish between node types and relation types, nor does it fully exploit edge features, spatial-temporal information and semantic information in heterogeneous multigraph.

The performance of LBSN2vec++ is much better than Walk2friend. This may be due to the fact that the hypergraph constructed by LBSN2vec++ contains more spatial-temporal and semantic information than the user-location bipartite graph constructed by Walk2friend. Moreover, the proposed walking strategy of random-walk-with-stay can obtain richer information than the general random walk.

Although both Heter-GCN and LBSN2vec++ construct heterogeneous graphs, the heterogeneous hypergraph constructed by LBSN2vec++ mainly focuses on the check-in relationship between users and POIs, and ignores the complex topology between users and POIs. The random walk strategy used by LBSN2vec++ does not capture this structural information effectively. Therefore, the performance of LBSN2vec++ is slightly worse compared to Heter-GCN.

HAN and MAGNN are more obviously improved than Heter-GCN. This is because the heterogeneous graph constructed by Heter-GCN using LBSN data is relatively simple, ignoring the rich spatial-temporal and semantic information in LBSN data. In contrast, HAN and MAGNN both use meta-paths, which are able to learn the higher-order interaction information of nodes. They also use the attention mechanism, which can better understand the importance of different nodes and meta-paths, thus improving the recommendation performance.

In general, compared with graph representation learning models (Walk2friend, Metapath2vec and LBSN2vec++), GNN-based models (Heter-GCN, HAN, MAGNN, and GEHMAN) perform better. It is because the latter

can utilize the rich information provided by the graph, such as the structural information of the graph. Compared with only utilizing the topological information in the social network, the full utilization of the rich semantic information in heterogeneous social networks can also significantly improve the friend link prediction performance.

Qualitative Analysis. In the comparison between the GEHMAN model and the other six models, in-depth analysis can be conducted from multiple dimensions, including graph model construction, node feature initialization, and feature learning.

First, regarding graph model construction, GEHMAN constructs a heterogeneous multi-graph, defining multiple relationships between users and POIs, including two node types and seven edge types. In contrast, Metapath2vec and Walk2Friends work on the basis of existing social networks or graph structures. Although other LBSN2vec++, MAGNN, Heter-GNN, and HAN have constructed graph models, the graph structures are relatively simple. This design of GEHMAN enables the model to effectively capture the complex associations between users and users, users and POIs, and POIs and POIs.

From the perspective of node feature initialization, the skip-gram and GRU models are used to analyze the user trajectory to learn the features of users and POIs and improve the accuracy of feature representation. LBSN2vec++, which also focuses on user trajectory data, is relatively simple in processing location features. Other models do not consider spatiotemporal information enough. In addition, GEHMAN, LBSN2vec++, Metapath2vec, and Walk2Friends all use skip-gram for embedding learning. However, Metapath2vec uses skip-gram to embed the random walk generated by the meta-path. MAGNN, Heter-GNN, and HAN do not directly use the skip-gram method.

In terms of feature learning, both HAN and GEHMAN models use hierarchical attention mechanisms to aggregate features of different types of nodes and edges, capturing relationships and features more effectively. However, GEHMAN does not rely on meta-paths, which can simplify the modeling process and achieve automated feature mining. And Heter-GCN learns node features through graph convolutional networks.

Overall, this multi-dimensional design gives GEHMAN strong expressiveness and flexibility, making it an important tool for processing LBSN.

Ablation Experiment

We choose two datasets, New York City (NYC) and Tokyo (TKY), as the ablation experiment datasets. This is because the size of the data in these two cities is representative and can well show the performance of different components of GEHMAN at different levels of data richness.

In order to analyze the effectiveness of the initial node feature learning of the GEHMAN model, the heterogeneous multigraph and attention mechanism in the node feature learning and the supervised comparative learning method in the model training, six variants of the GEHMAN model are designed in this section for comparison with the GEHMAN model. The definitions of these variants are as follows:

- GEHMAN-Init: Remove the node initial feature learning module. The node features of heterogeneous multigraph are initialized with random small values. The rest of the model remains unchanged.
- GEHMAN-Ae: Remove edge-level attention from node feature learning, leaving the importance of neighboring nodes the same. The rest of the model remains unchanged
- GEHMAN-As: Remove semantic-level attention from node feature learning, leaving the importance of each semantic representation the same. The rest of the model remains unchanged.
- GEHMAN-A: Remove the attention mechanism from node feature learning, leaving the importance of neighboring nodes and semantic representations the same. The rest of the model remains unchanged.
- GEHMAN-G: To transform the heterogeneous multigraph into a heterogeneous simple graph, we prioritize the retention of edges based on the following types: friendship edges, check-in edges, co-class edges, activity association edges, user co-occurrence edges, POI co-occurrence edges, and co-visit edges. This prioritization allows us to select which edges to keep while ensuring that each pair of nodes is connected by only one edge. As a result, we construct a heterogeneous simple graph. The rest of the model remains unchanged.
- GEHMAN-Cro: The supervised contrast learning is removed from model training and replaced with a cross-entropy loss function. The rest of the model remains unchanged.

The ablation experiment results are shown in Figure 11. Compared with GEHMAN, the performance of GEHMAN-A and GEHMAN-Cro on the three evaluation indicators has dropped significantly, with GEHMAN-A dropping by 3.05%, 3.41%, and 7.1% on average, and GEHMAN-Cro dropping by 3.26%, 3.5%, and 6.88% on average, respectively. This shows the effectiveness of the attention mechanism and supervised contrastive learning of the model. At the same time, from the comparative performance of GEHMAN-Ae, it can also be seen that edge-level attention is necessary for the performance of the model, that is, among all neighbor nodes connected by a type of edges, the importance of each neighbor node to the target node through this type of edges is different. Therefore, it is very necessary to use edge-level attention to learn the edge-level weight coefficients of different neighbor nodes and edges,

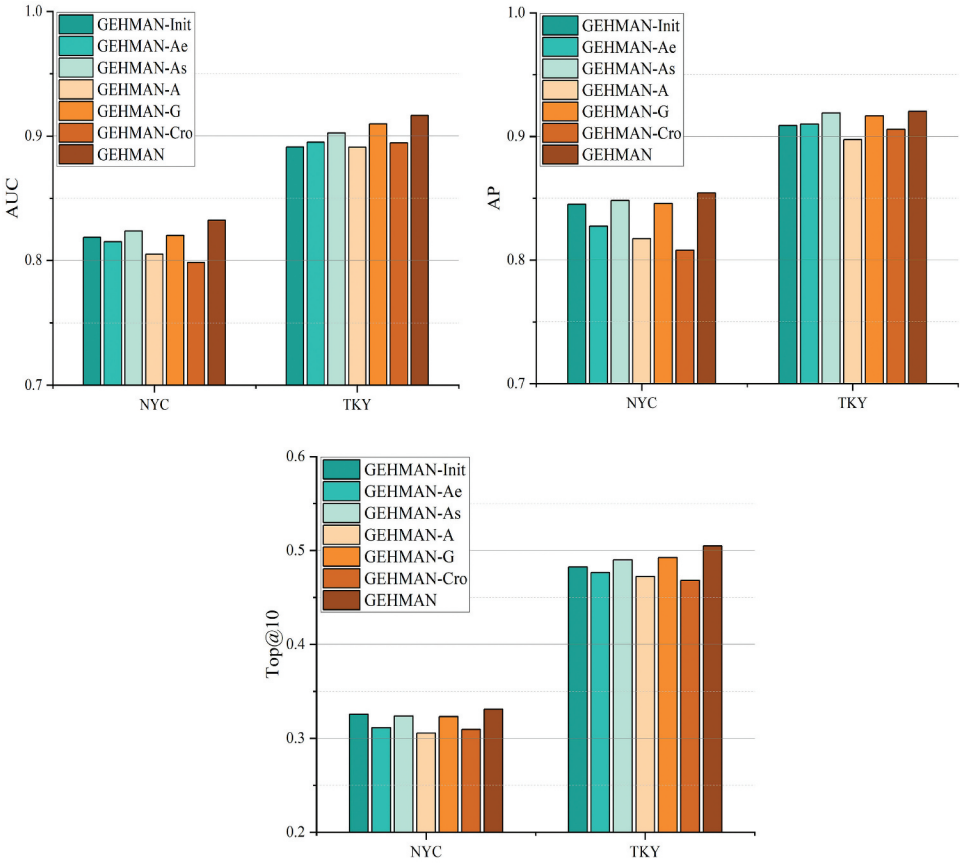

Figure 11. AUC, AP and Top@10 metrics for GEHMAN and its variants on NYC and TKY datasets.

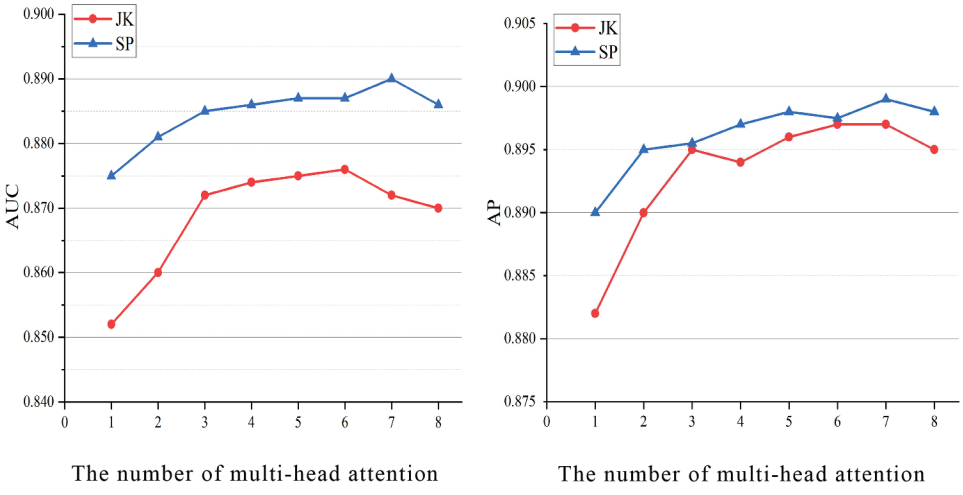

Figure 12. The impact of the number of multiple attention heads on the AUC and AP indicators on the JK and SP datasets.

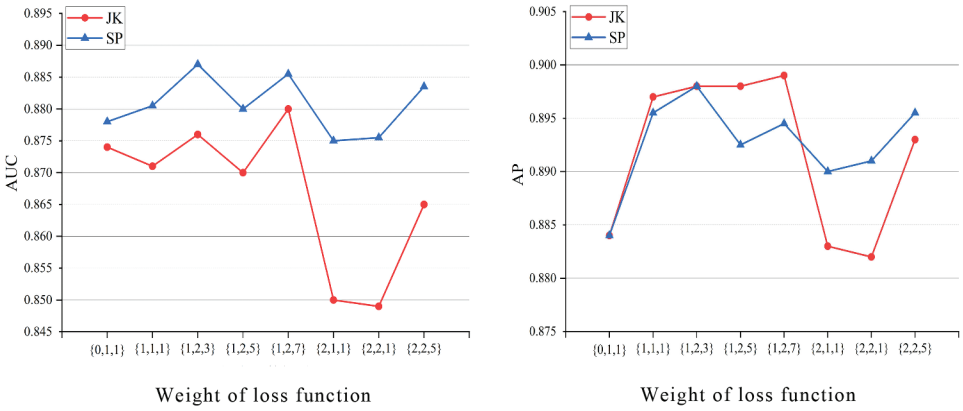

Figure 13. The impact of the loss function weight on the AUC and AP indicators on the JK and SP datasets.

rather than treating them equally. This helps the target node better integrate the information of neighbor nodes and edges.

Compared with the TKY datasets, the three indicators of GEHMAN-G perform worse on the NYC datasets. This may be because the NYC datasets is sparser than the TKY datasets. When using heterogeneous multigraph to model data, it is possible to model information such as spatial-temporal features more comprehensively than heterogeneous simple graph when the data is sparse. This shows that the construction of the heterogeneous multigraph and the additional design of several relationship feature edges have a positive impact on the GEHMAN model. However, this advantage is not obvious in data-rich datasets.

All three metrics of GEHMAN-IF decrease in both datasets. This shows that before node feature learning, the initial node features learned through skip-gram and GRU make the initial node features contain richer information, such as user movement trajectory features, so that the final representation of the user node is more accurate.

In conclusion, the ablation experiment demonstrates that the different components of the GEHMAN model are all indispensable and they can help the GEHMAN model to achieve better performance.

Related Experiment

Impact of Parameter Values on GEHMAN Performance. We study the impact of the number of multi-attention heads and loss function weights on evaluation indicators such as AUC and AP on the JK and SP city datasets. The experimental results are as follows.

As can be seen from figure 12, increasing the number of multi attention heads can make the model perform better, but it will also increase the usage of computing resources such as memory and GPU. When the number of multi-attention heads first increases, the model performance is greatly improved.

However, when the number of attention heads rises to a certain number, the performance of the model will remain stable within a certain range, and then as the number of attention heads increases, the model performance will even decline. Therefore, after comprehensive consideration, the value of the multi-attention heads of the model in our study is selected as 6.

The horizontal axis of the loss function weight in Figure 13 represents the value of the joint loss function λ. For example, $\lambda = \{0, 1, 1\}$ means $\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1$. Although the appropriate change of the weight of the joint loss function causes fluctuations in the AUC indicator, it can make the model have a better improvement in the AP indicator. However, if the weight is adjusted further, the convergence of the model will deteriorate. Therefore, after comprehensive consideration, the loss function weights of this research model are selected as $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3$.

The Average Performance of GEHMAN. We selected five negative samples in each training process and constructed the training set together with the positive samples. In order to comprehensively evaluate the performance of the model, we conducted multiple training and performed validation after every 10 epochs. During this process, we recorded the performance indicators of the model, including AUC, AP, and F1 scores, and calculated the average and standard deviation of these indicators, as shown in Figure 14.

In the bar chart, we can observe the average performance indicators of the GEHMAN model on six datasets (KL, SP, TKY, IST, JK, and NYC), where AUC and AP are both over 0.8. This result shows that the model performs very well on the classification task and can effectively distinguish between positive and negative samples. In addition, it reflects the strong generalization ability of the model, which means that the model maintains stable performance on unseen data.

F1 score analysis shows that except for IST and NYC datasets, the scores of other datasets are all above 0.7. This indicates that GEHMAN demonstrates a reasonable level of performance, striking a balance between precision and

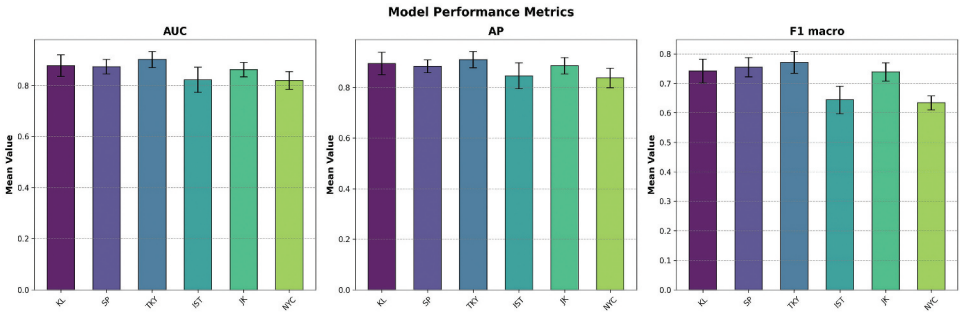

Figure 14. The mean and standard deviation of AUC, AP and F1-scores of the GEHMAN model.

recall. However, there is potential room for improvement in IST and NYC, perhaps because the complexity of IST and NYC pose challenges to the model.

In general, the bar chart clearly shows the good performance of the model on various datasets and emphasizes its effectiveness and application potential in the friend link prediction task.

Loss Curve. We have conducted a lot of experiments on six datasets, and the performance is very good. Take the performance on the SP and NYC datasets as examples.

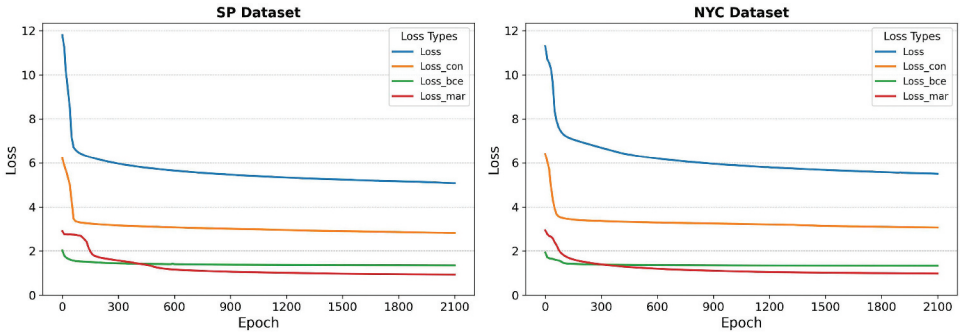

Figure 15. The curves of the joint loss function, contrast loss, binary cross entropy loss and margin loss changing with the training cycle (Epoch).

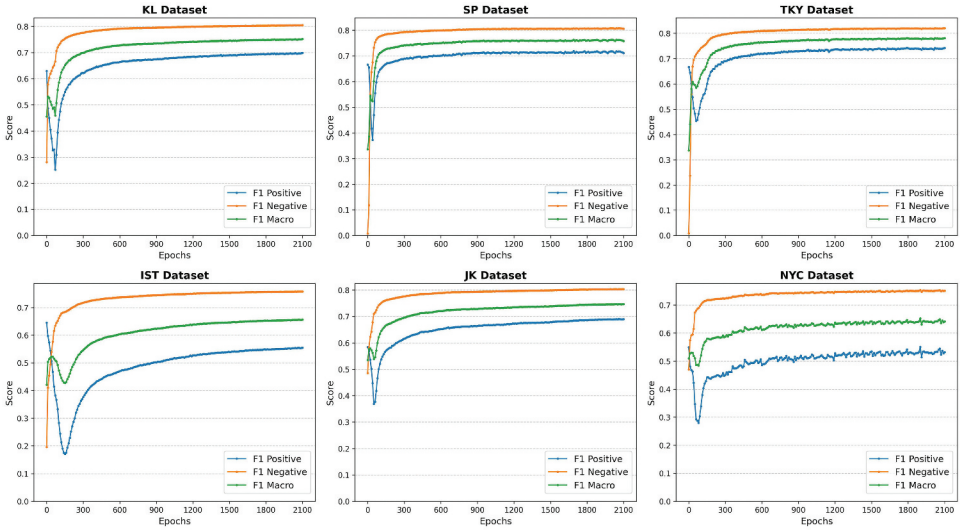

Figure 16. F1 scores for positive and negative classes and F1 macro for the GEHMAN model on six datasets.

Figure 15 shows the curves of the joint loss function, contrastive loss (Loss_con), binary cross entropy loss (Loss_bce) and margin loss (Loss_mar) changing with the training cycle (Epoch). It can be observed from the figure that all loss functions show a significant downward trend with the increase of training cycles, especially in the initial stage of training. It shows that the model quickly optimizes performance in early iterations. Among them, the joint loss and binary cross-entropy loss have a significant decrease during the training process, while the contrastive loss reaches a stable state earlier.

F1-Score. As mentioned earlier, we select five negative samples in each training and construct the training set together with the positive samples. The number of negative samples used overall is more than the number of positive samples. Therefore, the positive F1 score in Figure 16 is slightly worse at the beginning, while the negative F1 score rises faster.

The model has weak recognition ability for positive samples in the early stage of training. However, as the number of training rounds increases, the model is gradually able to better capture the characteristics of positive samples. Due to the large number of negative samples, the model can learn better features in the early stage, thereby achieving a higher F1 score in the early rounds. The F1-macro is the average of the positive and negative F1 scores. As the training progresses, it gradually stabilizes, indicating that the model has reached a balance point to a certain extent.

The F1 score converges at different speeds for different datasets (such as SP, TKY, and JK). It may be related to the characteristics of each dataset (such as category distribution, feature complexity, etc.).

Conclusion and Future Work

In view of the fact that previous work has not adequately explored and modeled the potential diverse associations in LBSN heterogeneous social networks, a model named GEHMAN based on heterogeneous multigraph and hierarchical attention is proposed for friend link prediction. The model constructs a user-POI heterogeneous multigraph to describe complex and diverse association relationships, fully utilizes the spatial-temporal information of user trajectories to provide meaningful input for GNN, adopts a hierarchical attention mechanism to update node information by fusing information of different types of neighbor nodes and connecting edges, and uses the idea of supervised comparative learning to train the model. Experiments on real datasets demonstrate that GEHMAN model is able to achieve effective deep mining of potential associations in LBSN and obtains better performance in friend link prediction compared to other benchmark methods.

Although the proposed model shows good results, there are still several important directions that deserve further research. First, integrating and effectively utilizing diverse user-generated content such as POI comments can provide deeper insights into user preferences and behaviors than friendship, check-in records, and POI data. Second, social networks are dynamic, and the network structure and attributes change over time. Methods that can adaptively learn user representations from dynamic graphs are needed. Addressing these multifaceted limitations brings opportunities for the development of friend recommendation and social network analysis.

Note

1. https://sites.google.com/site/yangdingqi/home/foursquare-dataset.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Aoxue Liu http://orcid.org/0009-0005-1335-8671
Yong Wang http://orcid.org/0000-0001-6261-8912

Data Availability Statement

The data that support the findings of this study are available in https://github.com/Liuaoxue/GEHMAN1/tree/master.

The datasets were derived from the following resources available in the public domain: https://sites.google.com/site/yangdingqi/home/foursquare-dataset.

References

Backes, M., M. Humbert, J. Pang, and Y. Zhang. 2017. walk2friends: Inferring social links from mobility profiles. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* 1943–57. doi: 10.1145/3133956.3133972.

Bahdanau, D., K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Berahmand, K., E. Nasiri, S. Forouzandeh, and Y. Li. 2022. A preference random walk algorithm for link prediction through mutual influence nodes in complex networks. *Journal of King Saud University-Computer & Information Sciences* 34 (8):5375–87. doi: 10.1016/j.jksuci.2021.05.006.

Bhatti, U. A., H. Tang, G. Wu, S. Marjan, and A. Hussain. 2023. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems* 2023 (1):8342104. doi: 10.1155/2023/8342104.

Bing, R., G. Yuan, M. Zhu, F. Meng, H. Ma, and S. Qiao. 2023. Heterogeneous graph neural networks analysis: A survey of techniques, evaluations and applications. *Artificial Intelligence Review* 56 (8):8003–42. doi: 10.1007/s10462-022-10375-2.

Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv Preprint arXiv:1312.6203*. doi: 10.48550/arXiv.1312.6203.

Chien, E., C. Pan, J. Peng, and O. Milenkovic. 2021. You are Allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*.

Cho, E., S. A. Myers, and J. Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* 1082–90. doi: 10.1145/2020408.2020579.

Daud, N. N., S. H. Ab Hamid, M. Saadoon, F. Sahran, and N. B. Anuar. 2020. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications* 166:102716. doi: 10.1016/j.jnca.2020.102716.

Dong, Y., N. V. Chawla, and A. Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* 135–44. doi: 10.1145/3097983.3098036.

Feng, Y., H. You, Z. Zhang, R. Ji, and Y. Gao. 2019. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (1):3558–65. doi: 10.1609/aaai.v33i01.33013558.

Fu, X., J. Zhang, Z. Meng, and I. King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. *Proceedings of the web conference 2020* 2331–41. doi: 10.1145/3366423.3380297.

Gao, Q., F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang. 2017. Identifying human mobility via trajectory embeddings. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* vol. 17, 1689–95.

Geng, X., Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (1):3656–63. doi: 10.1609/aaai.v33i01.33013656.

Girvan, M., and M. E. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99 (12):7821–26. doi: 10.1073/pnas.122653799.

Grover, A., and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* 855–64. doi: 10.1145/2939672.2939754.

Guo, J., W. Zhang, W. Fan, and W. Li. 2018. Combining geographical and social influences with deep learning for personalized point-of-interest recommendation. *Journal of Management Information Systems* 35 (4):1121–53. doi: 10.1080/07421222.2018.1523564.

Hamilton, W., Z. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* 30.

He, X., Z. He, J. Song, Z. Liu, Y. G. Jiang, and T. S. Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30 (12):2354–66. doi: 10.1109/TKDE.2018.2831682.

Li, J., H. Shomer, H. Mao, S. Zeng, Y. Ma, N. Shah, J. Tang, and D. Yin. 2024. Topology-aware uncertainty for image segmentation. *Advances in Neural Information Processing Systems* 36:8186–8207. doi: 10.5281/zenodo.6361906.

Li, Y., Z. Fan, D. Yin, R. Jiang, J. Deng, and X. Song. 2023. HMGCL: Heterogeneous multigraph contrastive learning for LBSN friend recommendation. *World Wide Web-Internet & Web Information Systems* 26 (4):1625–48. doi: 10.1007/s11280-022-01092-5.

Lin, L., J. Li, F. Chen, J. Ye, and J. Huai. 2017. Road traffic speed prediction: A probabilistic model fusing multi-source data. *IEEE Transactions on Knowledge and Data Engineering* 30 (7):1310–23. doi: 10.1109/TKDE.2017.2718525.

Liu, J., D. Djurdjanovic, J. Ni, N. Casoetto, and J. Lee. 2007. Similarity based method for manufacturing process performance prediction and diagnosis. *Computers in Industry* 58 (6):558–66. doi: 10.1016/j.compind.2006.12.004.

Ma, T., Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan. 2020. LGIEM: Global and local node influence-based community detection. *Future Generation Computer Systems* 105:533–46. doi: 10.1016/j.future.2019.12.022.

Ma, Z., and S. Nandy. 2023. Community detection with contextual multilayer networks. *IEEE Transactions on Information Theory* 69 (5):3203–39. doi: 10.1109/TIT.2023.3238352.

Park, K. G., and S. Han. 2018. How use of location-based social network (LBSN) services contributes to accumulation of social capital. *Social Indicators Research* 136 (1):379–96. doi: 10.1007/s11205-016-1525-9.

Perozzi, B., R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* 701–10. doi: 10.1145/2623330.2623732.

Samanta, S., V. K. Dubey, and B. Sarkar. 2021. Measure of influences in social networks. *Applied Soft Computing* 99:106858. doi: 10.1016/j.asoc.2020.106858.

Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20 (1):61–80. doi: 10.1109/TNN.2008.2005605.

Sun, Z., C. Li, Y. Lei, L. Zhang, J. Zhang, and S. Liang. 2021. Point-of-interest recommendation for users-businesses with uncertain check-ins. *IEEE Transactions on Knowledge and Data Engineering* 34 (12):5925–38. doi: 10.1109/TKDE.2021.3060818.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30.

Velickovic, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*. doi: 10.48550/arXiv.1710.10903.

Wang, X., H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. 2019. Heterogeneous graph attention network. *The world wide web conference* 2022–32. doi: 10.1145/3308558.3313562.

Wang, Z., T. Xia, R. Jiang, X. Liu, K. S. Kim, X. Song, and R. Shibasaki. 2021. Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks. *2021 IEEE 37th International Conference on Data Engineering* 1751–62. doi: 10.1109/ICDE51399.2021.00154.

Wu, H., C. Song, Y. Ge, and T. Ge. 2022. Link prediction on complex networks: An experimental survey. *Data Science and Engineering* 7 (3):253–78. doi: 10.1007/s41019-022-00188-2.

Wu, Y., D. Lian, S. Jin, and E. Chen. 2019. Graph convolutional networks on user mobility heterogeneous graphs for social relationship inference. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* 3898–904. doi: 10.24963/ijcai.2019/541.

Xie, Y., Z. Xu, J. Zhang, Z. Wang, and S. Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 45 (2):2412–29. doi: 10.1109/TPAMI.2022.3170559.

Yang, D., B. Qu, J. Yang, and P. Cudré-Mauroux. 2020. Lbsn2vec++: Heterogeneous hypergraph embedding for location-based social networks. *IEEE Transactions on Knowledge and Data Engineering* 34 (4):1843–55. doi: 10.1109/TKDE.2020.2997869.

Yang, D., B. Qu, J. Yang, and P. Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. *The world wide web conference* 2147–57. doi: 10.1145/3308558.3313635.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 1480–89 doi:10.18653/v1/n16-1174.

Yu, Y., H. Wang, and Z. Li. 2018. Inferring mobility relationship via graph embedding. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2 (3):1–21. doi: 10.1145/3264957.

Zhang, C., D. Song, C. Huang, A. Swami, and N. V. Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining*, 793–803. doi: 10.1145/3292500.3330961.

Zhang, W., X. Lai, and J. Wang. 2020. Social link inference via multiview matching network from spatiotemporal trajectories. *IEEE Transactions on Neural Networks and Learning Systems* 34 (4):1720–31. doi: 10.1109/TNNLS.2020.2986472.

Zou, M., Z. Gan, R. Cao, C. Guan, and S. Leng. 2023. Similarity-navigated graph neural networks for node classification. *Information Sciences* 633:41–69. doi: 10.1016/j.ins.2023.03.057.