Supplemental Material

We organize the supplementary material according to the structure and content of the main text: Appendix A presents challenges and their research status, Appendix B shows other data analysis results, Appendix C displays commonly used symbols, Appendix D details TGVx supplement, Appendix E gives experiments supplement, Appendix F introduces related work, and Appendix G gives references.

Appendix A Challenges and Their Research Status

(1) The interest drift issue caused by the spatiotemporal dynamics of user check-in behavior. Different from conventional recommendations as movies and news, the goal of the POI recommendation task is to provide the user with a venue in the physical world he/she likes, which is largely affected by various realistic factors [2,3,17]. POI recommendation is different from pure online interactions (e.g., movie recommendation), and its typical feature is that the user's check-in behavior has time [6,12,28,29] and spatial dynamics [3,4]. For example, users usually check in at a restaurant from 11:00 to13:00 and check in at a KTV from 19:00 to 21:00. When users visit in cities other than their hometown (e.g., business trip or tourism), their interests often change [3,4]. For example, few people in their hometown need to stay in a hotel, but they often need to check in to the hotel when they visit other cities. How to solve the issue of spatiotemporal interest drift is very challenging.

(2) Fusion of multi-source heterogeneous information alleviates data sparsity and cold start issues. Compared with recommendation tasks such as movies and music, POI recommendation faces more serious data sparsity and cold start issues. There are millions of POIs in an LBSN, but each user can only access a limited number of physical venues, especially

in the out-of-town recommendation scenario [2,3,4]. For example, the data density of Netflix for movie recommendation is about 1.2% [30], that of Foursquare and Yelp for POI recommendation is less than 0.5% [31,32], and the lower data density (less than 0.03%) for out-of-town recommendations [3,4]. A promising solution is to incorporate other types of information into the check-in data. For example, [14,33] integrates geographic information, [7,34] integrates social information, [6,15] integrates collaborative information, [3,4] integrates public and semantic information (e.g., categories, tags), and so on. However, multi-source information often has multi-modal heterogeneity. For example, geographic location coordinates are dense real values, while semantics are usually expressed as sparse word vectors. The more types of information that need to be fused, the more difficult it is, because the correlation between different modalities is highly non-linear [4].

(3) It is very difficult to model complex high-order nonlinear user-POI interactions with implicit feedback. Recommendations in the conventional fields mostly model user preferences based on explicit feedback, e.g., the ratings of movie recommendations. However, most POI recommender systems are based on implicit feedback, such as user check-in records [7,35]. Implicit feedback has no negative feedback. Observing the user's check-in behavior, it is difficult for us to reliably infer which POIs the user dislikes, only which POIs the user may like. The POIs that show that they have not checked in may be caused by the user's dislike, unawareness, and undisclosed for privacy protection, etc. [7,9,17]. The non-check-in information is also crucial because this is the easiest place to find negative feedback [36]. Modeling complex high-order nonlinear user-POI interactions based on implicit feedback is very challenging.

To solve the issue of interest drift that changes with time and space, some dynamic POI recommendation models have appeared [10,13,28]. Most of these studies are aimed at local users, focusing on the time dynamics of user check-in behavior, and solving the issue of interest drift over time. They assume that users' POI preferences are stable in different geographic areas/cities ignore the spatial dynamics of user interests and cannot provide high-quality POI recommendations for out-of-town users. Recently, some advanced POI recommendation models suitable for local and out-of-town scenarios have been published, e.g., UPS-CF [37], Geo-SAGE [38], ST-SAGE [39], ST-TransRec [5]. They pay attention to the spatial dynamics of user check-in behavior and solve the issue of interest drift that varies with geographic regions/cities. There are relatively few models that can provide time-dynamic POI recommendations for out-of-town users, and the representative ones are ST-LDA [3] and SH-CDL [4]. ST-LDA predicts the user's POI preferences based on the latent topic model, uses the time pattern of public check-in activities to mine topics, and finds that the time distribution of public check-in activities based on the daily mode can provide strong support for high-quality topics. SH-CDL uses public preferences with temporal characteristics to improve the quality of POI latent vector representation. ST-LDA and SH-CDL both "indirectly" affect the time factor in user-POI interaction. Whether it is providing POI recommendations to local users or out-of-town users, it should be personalized and spatiotemporal-aware [3]. This paper addresses **the first key issue** of how to realize the "direct" effect of the time factor on the user-POI interactions in the out-of-town recommendation scenario to better handle the spatiotemporal dynamics of interest drift.

To alleviate data sparsity and cold start issues, especially in out-of-town recommendation scenarios, a widely recognized strategy is to use auxiliary information related to check-in data, including: LBSN special factors (i.e., geographic, and temporal information) and social networks common factors (e.g., semantic, social, collaborative, and other information). The key to the successful implementation of this strategy lies in the effective mining of multi-source information and the reasonable fusion of multi-modal and heterogeneous information. Existing studies mostly follow the idea of Matrix Factorization (MF) [40], for example, Weighted-Regularized MF (WRMF) [33,34,41] and Probabilistic MF (PMF) [4,42,43] POI recommendation models. These studies are dedicated to fusing multi-source information into the user and POI latent vectors, and then obtaining the user's POI preference through the inner product operation of the vectors. At present, one of the most advanced fusion technologies is the Multimodal Deep Belief Network (MDBN) developed by Yin et al. [4]. It can fuse three heterogeneous information of geography, semantics, and popularity, and extract a unified semantic latent vector representation for each POI. In essence, these studies "indirectly" integrate multi-source heterogeneous information into user-POI interactions.

To find a more effective method, a small number of scholars began to explore the "direct" integration of multi-source heterogeneous information into user-POI interaction. For example, the Semi-CDAE model published by Guo et al. [7] in the JMIS journal and the SAE-NAD model published by Ma et al. [44] at the CIKM conference. The former "directly" integrates geographic, social, and POI category information into user-POI interactions through the visibility layer and conditional layer of Semi-RBM to obtain user POI preferences. The latter uses the Neighbor-Aware decoder to "directly" integrate geographic information into the user-POI interactions. The success of “direct” fusion is attributed to the multi-level and diversified structures of the Deep Learning (DL) [45] network, while “indirect” fusion is largely limited by the single-layer structure of the MF model. Unfortunately, neither Semi-CDAE nor SAE-NAD models involve time information. As a special factor of LBSN, time information has a crucial impact on the quality of POI recommendations [6]. Moreover, Semi-CDAE and SAE-NAD consider the geographical similarity of local users' check-ins to local POIs and ignore the spatial dynamics of user check-in behavior across cities and are not suitable for out-of-town recommendations. **The second key issue** addressed in this paper relates to the challenge of using multi-level and diversified structural advantages of DL network to “directly” fuse more types of heterogeneous information into user-POI interactions.

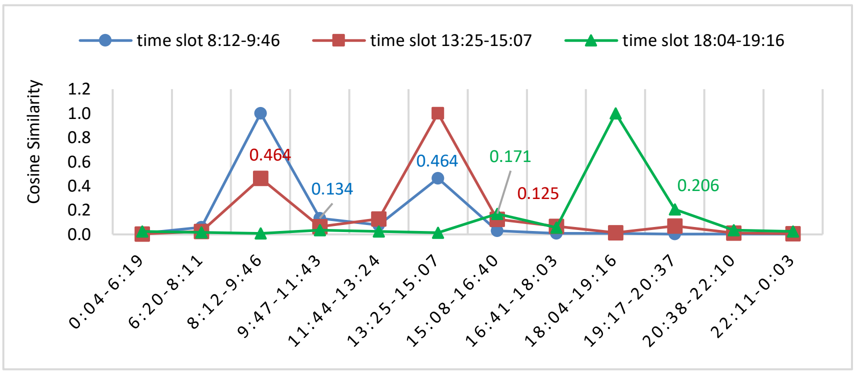
Due to its strong non-linear learning ability and high-order feature representation ability, DL technology has gradually emerged in the recommendation field [46,47]. For example, the MDBN technology developed by Yin et al. [4] is used to extract high-quality POI latent vectors, and Wang et al. [42] use Convolutional Neural Network (CNN) to extract image embedding vectors. However, the role of DL technology in the two POI recommendation models is not to dig deep into user-POI interactions, and the output of the DL network is not the user’s POI preference. Recently, some research has begun to use the power of DL technology to dig deeper into the complex high-order nonlinear interactions of user-POI [7,32,35,44], which we call the **deep-interaction POI recommendation** models. From the perspective of implicit feedback modeling, unsupervised learning has more advantages than supervised learning, because implicit feedback has no clear label information [36]. The biggest advantage of unsupervised learning is that it does not require label information and is dedicated to mining the internal relationships or rules of samples [45], e.g., user-POI interaction. However, existing deep-interaction POI recommendation models are purely static models, and do not consider the spatiotemporal dynamics of the user's check-in behavior. The recommendation list they generate for user u will not change with time, nor will it change with the regions/cities where u is located. **The third key issue** addressed in this paper is how to use implicit feedback to build a spatiotemporal dynamic deep-interaction POI recommendation model under the unsupervised learning paradigm.

Appendix B Other Data Analysis Results

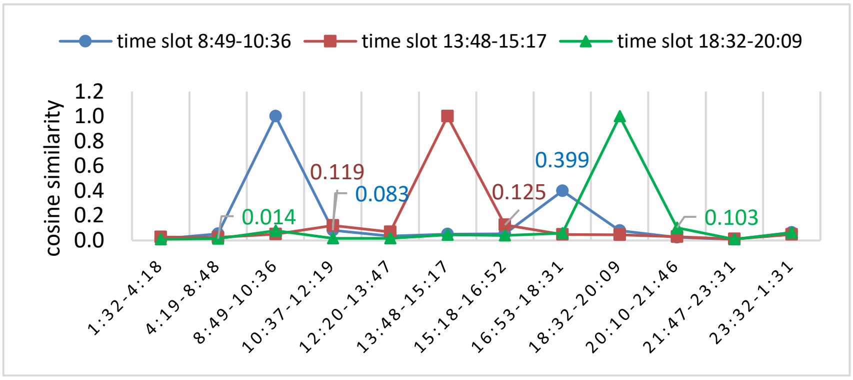
B.1 Figures and Tables in Section 2

Table 1 Basic information of the datasets.

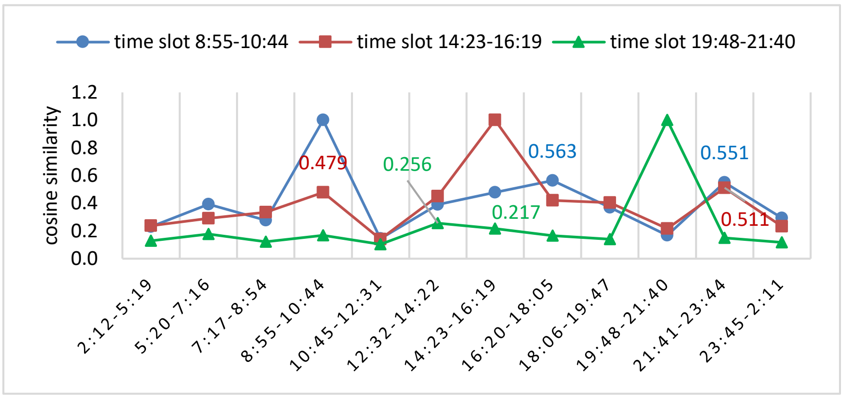
Target city: Tokyo												
Total number	Users: 6758		POIs: 4918		Checkins: 541926		Categories:291		Social friends		CF users	
w.r.t.	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out
number	4105	2653	4912	3062	498582	43344	286	194	34123	10398	17674	11453
V in home-town		Checkins: 166421			Checkin_POIs: 5109				Checkin_categories: 232			
Target city: Istanbul												
Total number	Users: 9562		POIs: 5819		Checkins: 657395		Categories: 302		Social friends		CF users	
w.r.t.	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out
number	6857	2705	5813	3019	626137	258	301	182	28169	7693	11496	8149
V in home-town		Checkins: 258370			Checkin_POIs: 3991				Checkin_categories: 198			
Target city: New York												
Total number	Users: 9283		POIs: 5831		Checkins: 196791		Categories: 299		Social friends		CF users	
w.r.t.	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out	loc	Out
number	4736	4547	5801	3549	170707	26084	293	216	41961	11365	20562	13786
V in home-town		Checkins: 98370			Checkin_POIs: 4043				Checkin_categories: 263			



(a) Tokyo



(b) Istanbul

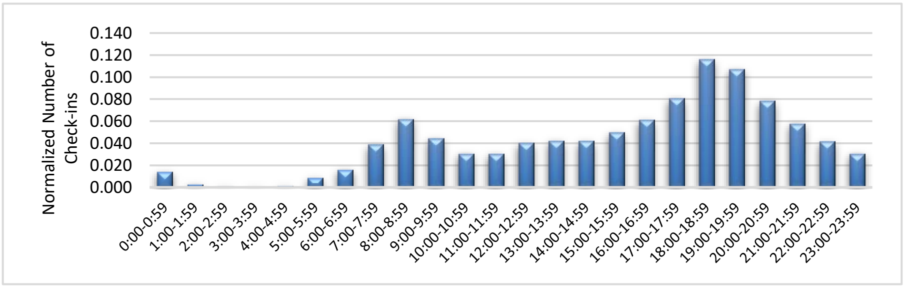


(c) New York

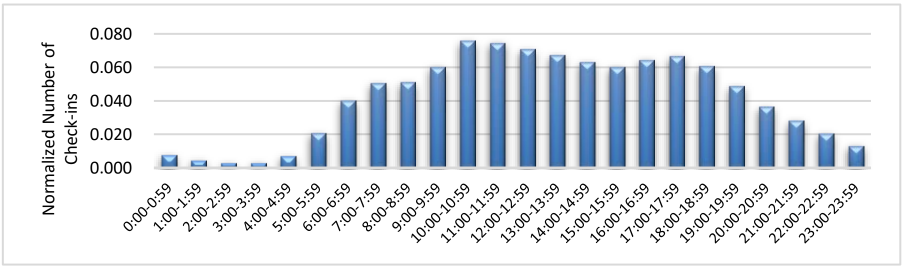
Fig.3 Curves of similarity of user check-in activities between unequal time slots.

B.2 Equal Time Slots

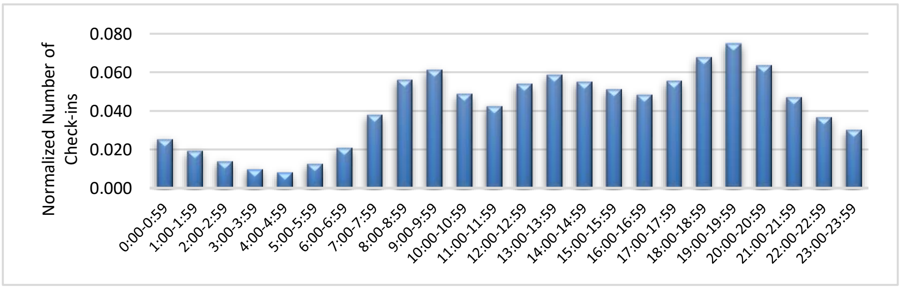
References [3,12] divided the check-in records into 24 equal time slots in hours. Fig.4 plots the time distribution of user check-in activities based on equal time slots [3], the vertical axis represents the standardized check-in number, and the horizontal axis is the time slot. From Fig.4, we observe that the check-in time distributions of the three cities have some **similarities**. For example, 1:00~5:59 users have fewer check-in activities (maybe sleeping at home), and most check-in activities are concentrated at 7:00~22:59. There are also some **differences** in the distribution of check-in times in the three cities. For example, Tokyo and Istanbul have two check-in peaks, Tokyo's two check-in peaks are 18:00~18:59 and 8:00~8:59; Istanbul's two check-in peaks are 10:00~10:59 and 17:00~17:59; New York has three check-in peaks, namely 9:00~9:59, 13:00~13:59, 19:00~19:59. Each of the three cities has a time slot when the number of check-ins is particularly small, but the length, starting point, and scarcity of this time slot are not the same. Tokyo is 1:00~4:59, Istanbul is 1:00~3:59, and New York is 3:00~4:49. This phenomenon is probably due to the differences in geographical locations and cultural customs of different cities.



(a) Tokyo

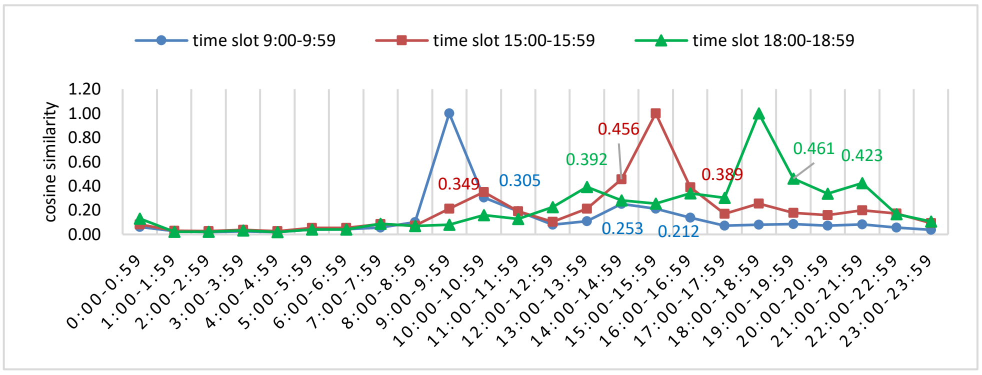


(b) Istanbul

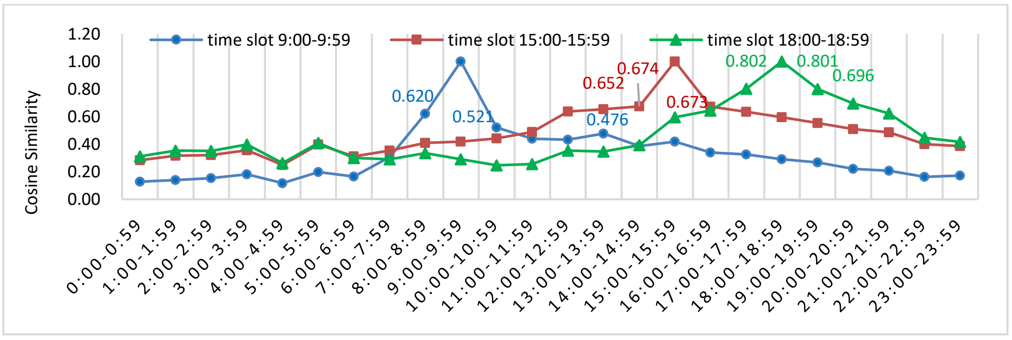


(c) New York

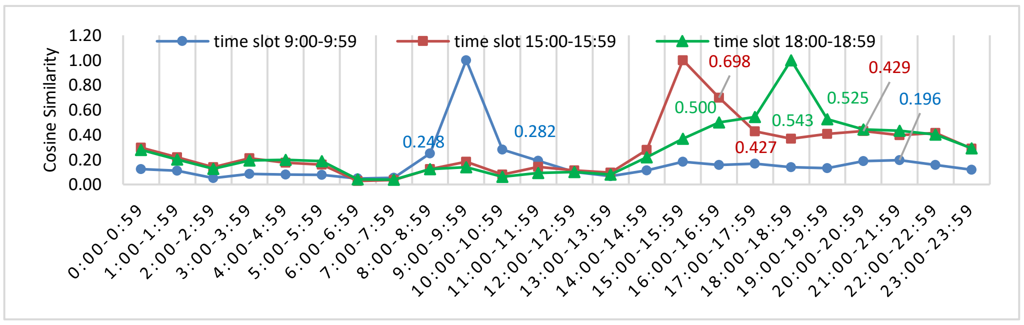
Fig.4 Time distribution of user check-in activities based on equal time slot.



(a) Tokyo



(b) Istanbul



(c) New York

Fig.5 Curves of similarity of user check-in activities between equal time slots

We also learn from the time pattern analysis method of Yuan et al. [6]. In Fig.5, the curves of the similarity of user check-in activities between equal time slots are drawn. The three curves represent three given time slots (i.e., 9:00~9:59, 15:00~15:59, 18:00~18:59), and each curve is marked with three largest similarities. Let $\mathcal{T}^{giv}$ denote a given time slot. It is not difficult to find from Fig.5 that the time slot most like $\mathcal{T}^{giv}$ is a certain adjacent time slot of $\mathcal{T}^{giv}$. This inspired us to appropriately increase the time granularity when dividing time slots, for example, to merge two adjacent time slots (i.e., two hours). In most cases, the second similar time slot to $\mathcal{T}^{giv}$ is another adjacent time slot of $\mathcal{T}^{giv}$, except for Tokyo's 9:00~9:59 and 18:00~18:59 and New York's 15:00~15:59. In many cases, the third similar time slot to $\mathcal{T}^{giv}$ is far away from $\mathcal{T}^{giv}$ and is not a "neighbor's neighbor", such as the three given time slots in Tokyo, 9:00~9:59 in Istanbul, and 9:00~9:59 and 15:00~15:59 in New York. This is also in line with the actual situation. For example, users often check in to the restaurant during lunch and dinner time.

B.3 Social Friends and Collaborative Users

Definition 1 (Social Friends). The social friends of user u refer to users who have a direct or potential social relationship with u (i.e., friends of friends) in a LBSN. $\boldsymbol{\mathcal{U}}_u^{soc}$ represents the set of u's social friends.

If the directionality of the friendship relationship (i.e., follow, be followed, follow each other) is not considered, the network formed by users and social relationships can be represented as an undirected graph, in which users with direct edge connections are called **explicit friends**. Friends often have some common interests, resulting in similar check-in behaviors [3,7,10,14]. There are also **potential friends** in social networks. That is, there is no direct edge connection between users, but it is likely to develop into explicit friends in the future. Potential friends also affect the user's check-in behavior. For example, users u_1 and u_2 are explicit friends, u_2 and u_3 are also explicit friends, but u_1 and u_3 are not explicit friends. If u_3 visited a good noodle restaurant and recommended it to u_2, it might be recommended to u_1 even if u_2 did not visit the noodle restaurant. This situation is the influence of friend's friend (i.e., u_3) of u_2 on user u_1, which is a potential friend influence, and u_3 is likely to become an explicit friend of u_1 in the future.

Definition 2 (Collaborative Users). Collaborative users of user u refer to users who have similar check-in behaviors to u in the LBSN but do not have a direct or potential social relationship. Collaborative users are also called similar users. $\mathcal{U}_u^{CF}$ represents the collaborative user set of u.

User-based Collaborative Filtering (CF) is one of the most classic recommendation methods. Its basic idea is that users with similar behaviors tend to have similar hobbies and are more likely to check in to the same POIs. This method predicts the POI preferences of u by aggregating the historical check-in behaviors of the collaborative users of user u [6,14]. Many studies and practical applications have verified the importance of collaborative information for recommendation [15].

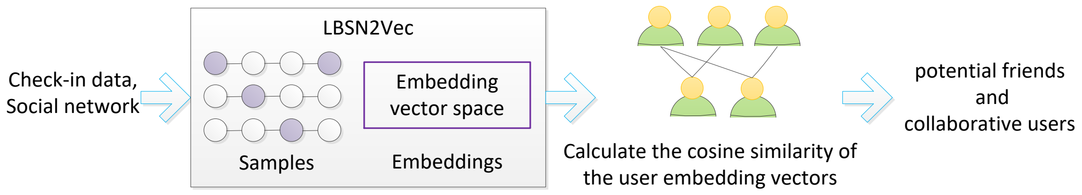


Fig.6 The mining process of the potential friends and collaborative users based on LBSN2vec.

LBSN2vec can combine social proximity and check-in behavior similarity to better predict potential friends and collaborative users. Yang et al. [13] described the LBSN2vec technology in detail and open sourced the code[1], so we only make a brief introduction. LBSN data can be regarded as a hypergraph, consisting of four kinds of nodes (i.e., user, POI category, time, POI), the social relationship edge connecting the user node pair, and the super edge connecting the check-in information which is user-category-time-POI. LBSN2vec learns nodes embeddings from the social relationship edges and check-in super edges of the LBSN hypergraph, and

[1] https://github.com/eXascaleInfolab/LBSN2Vec

retains the key structural attributes of the graph, which can effectively perform graph analysis tasks (e.g., social relationship prediction).

Fig.6 displays a schematic diagram of the potential friends mining process based on LBSN2vec. First, the explicit friend relationship and check-in information are jointly sampled from the LBSN hypergraph, and the embedding matrix is obtained by learning node embedding. Then, the cosine similarity between the user node and the embedding vector in the embedding matrix is calculated. When the cosine similarity is greater than the threshold $\emptyset(0 \leq \emptyset \leq 1)$, the user relationship is defined as a potential friend relationship.

B.2.1 Random Walk with Stay

LBSN2vec uses a stay-able random walk method for sampling. First, a classic random walk algorithm is performed on the side of the user's social relationship. Then, for each user node encountered, stay on the user node to sample the user's check-in information. The node embedding learning process alternates between these two types of edges. The influence range of social relations and user check-in information is specified by setting an adjustable parameter to control the edge type of sampling.

B.2.2 Embedding

A set of nodes $\{x_i\}$ are sampled from the LBSN hypergraph. By maximizing the cosine similarity between each node embedding vector $\vec{x}_i$ and the best fit line $\vec{x}_{best}$ in the embedding matrix, the proximity θ between nodes x_i and $\vec{x}_{best}$ is optimized:

$$\theta = \sum_{i=1}^{|Nodes|} cos(\vec{x}_i, \vec{x}_{best}), \tag{A1}$$

where $|Nodes|$ stands for the total number of nodes. $\vec{x}_{best}$ is calculated by Eq.(A2):

$$\vec{x}_{best} = \sum_{i=1}^{|Nodes|} \frac{\vec{x}_i}{\|\vec{x}_i\|}, \tag{A2}$$

Moreover, a negative sampling technique is applied to accelerate model training. The cosine distance between each negative sample node x_{neg} and the best fit line vector $\vec{x}_{best}$ is Maximized. The final objective function becomes:

$$\theta = \sum_{i=1}^{|Nodes|} \left(cos(\vec{x}_i, \vec{x}_{best}) + \gamma \mathbb{E}_{x_{neg}} \left[1 - cos\left(\vec{x}_{neg}, \vec{x}_{best}\right) \right] \right), \tag{A3}$$

where γ is the number of negative samples.

Stochastic gradient descent algorithm is applied to optimize the objective function. During the node learning process, $\vec{x}_{best}$ is fixed, and only the gradients of $\vec{x}_i$ and $\vec{x}_{neg}$ need to be calculated:

$$\frac{\partial \theta}{\partial \vec{x}_i} = \frac{\vec{x}_{best}}{\|\vec{x}_i\| \|\vec{x}_{best}\|} - \frac{\vec{x}_i cos(\vec{x}_i, \vec{x}_{best})}{\|\vec{x}_i\|^2}, \tag{A4}$$

$$\frac{\partial \theta}{\partial \vec{x}_{neg}} = \frac{\vec{x}_{best}}{\|\vec{x}_{neg}\| \|\vec{x}_{best}\|} - \frac{\vec{x}_{neg} cos(\vec{x}_{neg}, \vec{x}_{best})}{\|\vec{x}_{neg}\|^2}. \tag{A5}$$

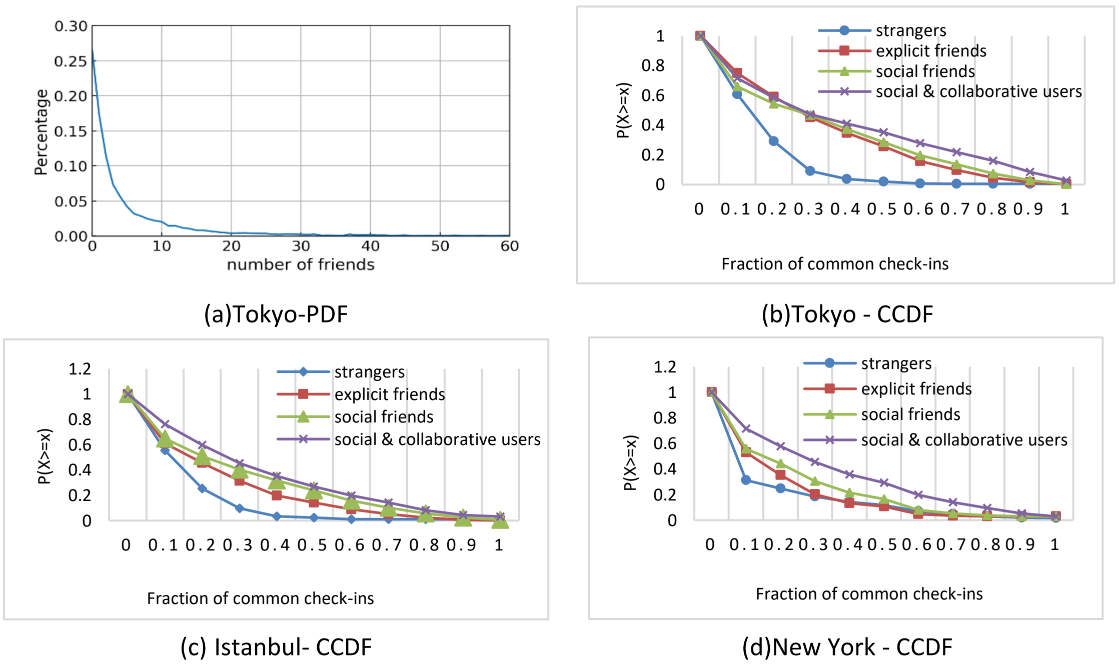


Fig.7(a) PDF curve of the number of explicit friends, and (b~d) CCDF curves of users' common check-in ratio

Appendix C Commonly Used Symbols

Table 2 Commonly used symbols of the paper

Symbol	Description	Symbol	Description
$\mathcal{u}, \boldsymbol{\mathcal{u}}$	a user and the latent vector of $\mathcal{u}$	$\mathcal{v}, \boldsymbol{\mathcal{v}}$	a POI and the latent vector of $\mathcal{v}$
c	a POI category	$\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{C}}$	user set, POIs set, POI category set
$\lvert\cdot\rvert$	the number of elements in a set	$\mathcal{N}.$	quantity
$\mathcal{p}_{\mathcal{u}\mathcal{v}}$	$\mathcal{u}$'s preference for $\mathcal{v}$	$\boldsymbol{\mathcal{p}}_{\mathcal{u}}$	$\mathcal{u}$'s POI preference vector in the target city
t	check-in time	$\mathcal{T}^{giv}$,$\mathcal{T}^{sim}$	given and similar time slots
$\mathbb{V}_i, \mathbb{H}_j$	visible and hidden units	a_i, b_j	biases of $\mathbb{V}_i$ and $\mathbb{H}_j$
w_{ij},	connection weights of $\mathbb{V}_i$ and $\mathbb{H}_j$	$\mathbf{W}^r$	weight matrix of (Semi)RBM#r r=0,1,…,2R-1
$\mathbb{T}_i^n, \mathbb{T}^n$	temporal condition unit and layer	$\beta_{ii}^n, \alpha_{ij}^n$	connection weights of $\mathbb{T}_i^n$&$\mathbb{V}_i$, and $\mathbb{T}_i^n$&$\mathbb{H}_j$
$\mathbf{s}^{vis}$	visible layer state vector, $s_i \in \mathbf{s}^{vis}$	$\mathbf{s}_{-i}^{vis}$	visible layer state vector without s_i
$\mathbf{s}^{hid}$	hidden layer state vector,$h_j \in \mathbf{s}^{hid}$	$\mathbf{t}^{sim_n}$	$\mathbb{T}^n$ state vector, $t_i^{sim_n} \in \mathbf{t}^{sim_n}$
γ_{iz}	connection weight of $\mathbb{V}_i$ and $\mathbb{V}_z$	$\mu^{\mathbb{V}_i}$	mean of s_i
$\boldsymbol{\beta}^n$	connection weight vector of $\mathbb{T}^n$ and visible layer	$\boldsymbol{\alpha}^n$	fully connected matrix of $\mathbb{T}^n$and hidden layer

Appendix D TGVx Supplement

D.1 RBM

D.1.1 RBM#1~RBM#R-2 with Binary Units

SemiDAE's RBM#1~RBM#R-2 are RBMs with binary units. $\mathbf{s}^{vis}$ and $\mathbf{s}^{hid}$ represent the state vector of the visible layer and the hidden layer, respectively. The energy function of conventional RBM is [48]:

$$E\left(\mathbf{s}^{vis}, \mathbf{s}^{hid}\right) = -\sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} a_i s_i - \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} b_j h_j - \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} s_i h_j w_{ij}, \tag{A6}$$

where w_{ij} is the connection weight of the visible unit $\mathbb{V}_i$ and the hidden unit $\mathbb{H}_j$. s_i and h_j are the binary states of $\mathbb{V}_i$ and $\mathbb{H}_j$, respectively. a_i and b_j are the bias of $\mathbb{V}_i$ and $\mathbb{H}_j$, respectively. $\mathcal{N}_{\mathbb{V}}$ and $\mathcal{N}_{\mathbb{H}}$ represent the number of visible units and hidden units.

According to the visible layer state vector $\mathbf{s}^{vis}$, the activation probability of the binary hidden unit $\mathbb{H}_j$ is [7,48]:

$$P\left(h_j = 1 \middle| \mathbf{s}^{vis}\right) = ReLU\left(b_j + \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} s_i w_{ij}\right), \tag{A7}$$

where $ReLU(\cdot)$ is the activation function. The study [49] found that to get the same training error, $ReLU(x)$ consumes less training time than other activation functions such as $Sigmoid(x)$.

According to the hidden layer state vector $\mathbf{s}^{hid}$, the activation probability of the reconstructed binary visible unit $\mathbb{V}_i$ is [7,48]:

$$P\left(s_i = 1 \middle| \mathbf{s}^{hid}\right) = ReLU\left(a_i + \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} h_j w_{ij}\right), \tag{A8}$$

The update rule (i.e., gradient) for obtaining the parameters w_{ij}, a_i, b_j based on the CD [23] algorithm is [48]:

$$\Delta w_{ij} = \varepsilon\left(\langle s_i h_j \rangle_{dat} - \langle s_i h_j \rangle_{rec}\right) \tag{A9}$$

$$\Delta a_i = \varepsilon(\langle s_i \rangle_{dat} - \langle s_i \rangle_{rec}) \tag{A10}$$

$$\Delta b_j = \varepsilon\left(\langle h_j \rangle_{dat} - \langle h_j \rangle_{rec}\right) \tag{A11}$$

where $\langle \cdot \rangle_{dat}$ is the expectation about the distribution of data. $\langle \cdot \rangle_{rec}$ is the expectation about the distribution of reconstructed data. ε is the learning rate. For more details on conventional RBM, please refer to [48].

D.1.2 RBM#R-1 with Gaussian Hidden Units

SemiDAE's RBM#R-1 is an RBM with real-valued Gaussian hidden units. That is, the binary hidden unit is replaced with a linear unit with independent Gaussian noise. The energy function becomes:

$$E\left(\mathbf{s}^{vis}, \mathbf{s}^{hid}\right) = \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} a_i\, s_i - \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} \frac{(h_j - b_j)^2}{2\sigma_j^2} - \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} s_i \frac{h_j}{\sigma_j} w_{ij}, \tag{A12}$$

where σ_j is the standard deviation of Gaussian noise in the real state h_j. In real-value applications, σ_j is set to 1, and the hidden unit state value h_j is reconstructed to make it obey the Gaussian distribution $\mathbb{N}\left(\mu^{\mathbb{H}_j}, 1\right)$. Therefore, the hidden unit $\mathbb{H}_j$ is sampled in the Gaussian distribution $\mathbb{N}\left(\mu^{\mathbb{H}_j}, 1\right)$ instead of Eq.(A7). Use the visible layer state vector $\mathbf{s}^{vis}$

and bias b_j to make a linear contribution to the mean value $\mu^{\mathbb{H}_j}$ of the real-valued state. Then reconstruct the activation probability of the real-valued hidden unit $\mathbb{H}_j$ in the Gaussian distribution $\mathbb{N}(\mu^{\mathbb{H}_j}, 1)$:

$$P(h_j|\mathbf{s}^{vis}) = \mathbb{N}(b_j + \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} s_i w_{ij}, 1), \tag{A13}$$

According to the hidden layer real-valued state vector $\mathbf{s}^{hid}$, the activation probability of the reconstructed binary visible unit $\mathbb{V}_\ell$ is still Eq.(A8). The parameter update rules of the RBM with real-valued Gaussian hidden unit is still Eq.(A9)~(A11).

Algorithm 1 The pre-training algorithm of RBM#$r, r = 1,2, \dots, R-1$

```
Input     Training set $\Xi$, hyperparameters, randomly initialized $\mathbf{W}^r, \boldsymbol{a}^r, \boldsymbol{b}^r$
Output    $\widehat{\mathbf{W}}^r, \widehat{\boldsymbol{a}}^r, \widehat{\boldsymbol{b}}^r$
step 1    Repeat until convergence
step 2      Shuffle $\Xi$
step 3      For each $\mathscr{b}$ in $\Xi$ :
step 4        $\mathbf{s}^{vis} = \mathscr{b}$
step 5        For $m = 0,1 \dots, \mathscr{k} - 1$ :
                  Sampling $\mathbb{H}_j, j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$:
step 6            The RBM of the binary hidden unit obtains $h_j^{(m)}$ according to Eq.(A7).
                  The RBM of the real-valued hidden unit obtains $h_j^{(m)}$ according to Eq.(A13).
Step 7            Reconstructing $\mathbb{V}_i, i = 1,2, \dots, \mathcal{N}_{\mathbb{V}}$: Obtain $s_i^{(m+1)}$according to Eq.(A8).
Step 8        For $j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$ :
                  Sampling $\mathbb{H}_j$:
                  The RBM of the binary hidden unit obtains $h_j^{(\mathscr{k})}$ according to Eq.(A7).
                  The RBM of the real-valued hidden unit obtains $h_j^{(\mathscr{k})}$ according to Eq.(A13).
Step 9        For $i = 1,2, \dots, \mathcal{N}_{\mathbb{V}}, j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$:
                  According to Eq.(A9)~(A11), update the parameters $\widehat{w}_{ij}, \widehat{a}_i, \widehat{b}_j$.
```

Refer to Algorithm 1 for the pre-training algorithm of RBM1~RBM#R-1. The training set Ξ is constructed using the state vector of the layer before the RBM#r's visible layer in SemiDAE as a sample. The specific values of $\mathcal{N}_{\mathbb{V}}$ and $\mathcal{N}_{\mathbb{H}}$ should be determined according to the RBM number. The number of steps in the CD algorithm is $\mathscr{k}$. Feed the visible layer one sample at a time (step 3~4). Sampling the hidden unit according to the state of the visible layer (step 6): The hidden layer is a binary unit, using Eq.(A7); the hidden layer is a real-valued unit, using Eq.(A13). After obtaining the hidden layer, reconstruct the visible unit (step 7). The above steps are repeated $\mathscr{k}-1$ times ($\mathscr{k}$=1 in most cases) (steps 5-7). Because the $\mathscr{k}$th sampling of the hidden layer is missing, the visible layer obtained in step $\mathscr{k}$ will be used to sample the hidden unit for the last time (step 8). After that, the update rules are obtained based on the CD algorithm, and the parameters are updated (step 9). Training can be repeated for multiple rounds as needed until convergence (steps 1-9), and shuffle the training set (step 2) before each round of training. Finally, the weight and bias estimates $\widehat{\mathbf{W}}^r, \widehat{\boldsymbol{a}}^r, \widehat{\boldsymbol{b}}^r$ of RBM#r are obtained.

D.1.3 SemiRBM#0 with Gaussian Visible Units

The energy function of the RBM with Gaussian visible units is [7,48]:

$$E(\mathbf{s}^{vis}, \mathbf{s}^{hid}) = \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \frac{(s_i-a_i)^2}{2\delta_i^2} - \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} b_j h_j - \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} \frac{s_i}{\delta_i} h_j w_{ij}, \qquad \text{(A14)}$$

where δ_i is the Gaussian noise standard deviation of the real-valued state s_i.

SemiRBM is an extended form of RBM, in which units within the visible layer and/or hidden layer are fully or partially connected. SemiDAE's SemiRBM#0 has a real-valued Gaussian visible unit, and its energy function is:

$$E(\mathbf{s}^{vis}, \mathbf{s}^{hid}) = \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \frac{(s_i-a_i)^2}{2\delta_i^2} - \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} b_j h_j - \sum_{i=1}^{\mathcal{N}_{\mathbb{V}}} \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} \frac{s_i}{\delta_i} h_j w_{\ell j} - \sum_{s=1}^{\mathcal{N}_{\mathbb{V}}} \sum_{z=1, z\neq i}^{\mathcal{N}_{\mathbb{V}}} \frac{s_i}{\delta_i} \frac{s_z}{\delta_z} \gamma_{iz} \qquad \text{(A15)}$$

where γ_{iz} is the connection weight of the visible units $\mathbb{V}_i$ and $\mathbb{V}_z$. The number of visible units $\mathcal{N}_{\mathbb{V}}$ of SemiRBM#0 is equal to $|\boldsymbol{\mathcal{V}}^{loc}|$, and the number of hidden units $\mathcal{N}_{\mathbb{H}}$ is less than $|\boldsymbol{\mathcal{V}}^{loc}|$.

According to the visible layer state vector $\mathbf{s}^{vis}$, the activation probability of the binary hidden unit $\mathbb{H}_j$ is still Eq.(A7). According to $\mathbf{s}^{hid}$ and $\mathbf{s}_{-i}^{vis}$, the visible unit $\mathbb{V}_i$ is reconstructed in the Gaussian distribution $\mathbb{N}(\mu^{\mathbb{V}_i}, 1)$ by Eq.(A16), instead of Eq.(A8). Specifically, first, the input data (i.e., the POI preference vector $\boldsymbol{\mathcal{p}}_{u}^{giv}$ of the user u on $\mathcal{T}^{giv}$) is subjected to z-score standardization processing to obtain $\breve{\boldsymbol{\mathcal{p}}}_{u}^{giv}$. Then, the δ_i and δ_z of Eq.(A15) are set to 1, and the real-valued state v_ℓ of the visible unit $\mathbb{V}_i$ is reconstructed to obey the Gaussian distribution $\mathbb{N}(\mu^{\mathbb{V}_i}, 1)$. Using the hidden layer state vector $\mathbf{s}^{hid}$, the visible layer state vector $\mathbf{s}_{-i}^{vis}$ that does not contain the state value s_i, and the bias a_i, make a linear contribution to the mean value $\mu^{\mathbb{V}_i}$ of s_i. Then, reconstruct the activation probability of the real-valued visible unit $\mathbb{V}_i$ in the Gaussian distribution $\mathbb{N}(\mu^{\mathbb{V}_i}, 1)$:

$$P\left(s_i \middle| \mathbf{s}^{hid}, \mathbf{s}_{-i}^{vis}\right) = \mathbb{N}\left(a_i + \sum_{j=1}^{\mathcal{N}_{\mathbb{H}}} h_j w_{ij} + \sum_{z=1, z\neq i}^{\mathcal{N}_{\mathbb{V}}} s_z \gamma_{iz}, 1\right), \tag{A16}$$

Algorithm 2 The traditional pre-training algorithm of SemiRBM#0

```
Input    Training set $\mathbf{\Theta}^{giv} = \{\breve{\boldsymbol{\mathcal{p}}}_u^{giv}: u \in \boldsymbol{\mathcal{U}}^{giv}\}$, hyperparameters, randomly initialized $\mathbf{W}^0, \boldsymbol{a}^0, \boldsymbol{b}^0$
Output   $\widehat{\mathbf{W}}^0, \widehat{\boldsymbol{a}}^0, \widehat{\boldsymbol{b}}^0$
step 1   **Repeat** until convergence
step 2     Shuffle $\mathbf{\Theta}^{giv}$
step 3     **For each** $\breve{\boldsymbol{\mathcal{p}}}_u^{giv}$ **in** $\mathbf{\Theta}^{giv}$ :
step 4         $\mathbf{s}^{vis} = \breve{\boldsymbol{\mathcal{p}}}_u^{giv}$
step 5         **For** $m = 0,1 \dots, \mathcal{k} - 1$ :
                   Sampling $\mathbb{H}_j, j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$: Obtain the state value $h_j^{(m)}$ according to Eq.(A7).
                   Reconstructing $\mathbb{V}_i, i = 1,2, \dots, \mathcal{N}_{\mathbb{V}}$: Obtain the state value $s_i^{(m+1)}$ according to Eq.(A16).
Step 6         **For** $j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$:
                   Sampling $\mathbb{H}_j$: Obtain the state value $h_j^{(\mathcal{k})}$ according to Eq.(A7).
Step 7         **For** $i = 1,2, \dots, \mathcal{N}_{\mathbb{V}}, j = 1,2, \dots, \mathcal{N}_{\mathbb{H}}$ :
                   According to Eq.(A9)~(A11), update the parameters $\widehat{w}_{ij}, \widehat{a}_i, \widehat{b}_j$。
```

When pre-training SemiRBM#0, update the parameters $w_{ij}, a_i, b_j, i = 1,2, \cdots, \mathcal{N}_{\mathbb{V}}, j = 1,2, \cdots, \mathcal{N}_{\mathbb{H}}$ based on the CD algorithm, and the update rule is still Eq.(A9)~(A11). The weight

γ_{iz} is calculated by Eq.(1) without training. For the traditional pre-training algorithm of SemiRBM#0, please refer to Algorithm 2, and its process is similar to Algorithm 1. For a given time slot $\mathcal{T}^{giv}$, the user set is $\boldsymbol{\mathcal{U}}^{giv}, \boldsymbol{\mathcal{U}}^{giv} \subset \boldsymbol{\mathcal{U}}$, and the training set is $\boldsymbol{\Theta}^{giv} = \{\breve{\boldsymbol{p}}_{u}^{giv} : u \in \boldsymbol{\mathcal{U}}^{giv}\}$. $\breve{\boldsymbol{p}}_{u}^{giv}$ represents $\boldsymbol{p}_{u}^{giv}$ standardized by z-score.

Because it is impossible for any user to check in all POIs, $\breve{\boldsymbol{p}}_{u}^{giv}$ will have many missing values "nan". When a visible unit receives a missing value, the connection weights between the visible unit and the hidden layer are frozen, and only the visible units that receive the non-missing value are reconstructed [48,52]. Different users have different POI check-in sets, corresponding to different visible unit sets. Even if a visible unit is not trained by the sample of user u, it will be trained by samples of other users' [7].

D.1.4 Conventional Pre-Training Algorithm for T-SemiRBM#0

When pre-training T-SemiRBM#0, the parameters $w_{ij}, a_i, b_j, \alpha_{ij}^{n}, \beta_{ii}^{n}, i = 1,2,\cdots,|\boldsymbol{\mathcal{V}}^{loc}|$, $j = 1,2,\cdots,\mathcal{N}_{\mathbb{H}}, n = 1,2,\cdots,q$ are learned and updated using the CD algorithm [23]. Eq.(A17) gives the update rules $\Delta\alpha_{ij}^{n}, \Delta\beta_{ii}^{n}$:

$$\begin{cases} \Delta\alpha_{ij}^{n} = \varepsilon\left(\langle h_j\rangle_{dat} - \langle h_j\rangle_{rec}\right) \cdot p_{ui}^{sim_n} \\ \Delta\beta_{ii}^{n} = \varepsilon(\langle s_i\rangle_{dat} - \langle s_i\rangle_{rec}) \cdot p_{ui}^{sim_n} \end{cases}, \tag{A17}$$

When $p_{ui}^{sim_n}$ is a missing value, α_{ij}^{n} and β_{ii}^{n} are not updated, that is, directly set $\Delta\alpha_{ij}^{n}$=0, $\Delta\beta_{ii}^{n}$=0.

See Algorithm 3 for the conventional pre-training algorithm of T-SemiRBM#0. For a given time slot $\mathcal{T}^{giv}$, the training set is $\boldsymbol{\Theta}^{giv} = \left\{\left(\breve{\boldsymbol{p}}_{u}^{giv}, \breve{\boldsymbol{p}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{p}}_{u}^{sim_q}\right) : u \in \boldsymbol{\mathcal{U}}^{giv}\right\}$. Each time a user sample is extracted from $\boldsymbol{\Theta}^{giv}$ and fed to the real-valued visible layer and the time-conditional network (steps 3~4). Then, the hidden units are sampled based on the visible layer and the time-conditional network (step 6). After obtaining the hidden layer, combine the geographical similarity between POIs to reconstruct the visible unit (step 7). The above steps are repeated $\mathscr{k}$-1 times ($\mathscr{k}$=1 in most cases) (steps 5-7). Because the $\mathscr{k}$th sampling of the hidden layer is missing, the visible layer obtained in step $\mathscr{k}$ will be used to sample the hidden unit for the last time (step 8). After that, the update rule is obtained based on the CD algorithm and the parameters are updated (step 9). Repeat training for multiple rounds as needed until convergence (steps 1-9), and perform shuffle operation on $\boldsymbol{\Theta}^{cur}$ (step 2) before each round of training. Finally, the pre-estimated value $\widehat{\mathbf{W}}^0, \widehat{\boldsymbol{a}}^0, \widehat{\boldsymbol{b}}^0, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}$ of the weights and biases of T-SemiRBM are obtained. During the training process, when the time-condition unit $\mathbb{T}_i^n$ receives a missing value, the connection weight between $\mathbb{T}_i^n$ and the visible unit $\mathbb{T}_i^n$ and the fully connected weights between $\mathbb{T}_i^n$ and the hidden layer are not updated.

Algorithm 3 The conventional pre-training algorithm of T-SemiRBM#0

```
Input   Training set $\boldsymbol{\Theta}^{giv} = \left\{\left(\breve{\boldsymbol{p}}_u^{giv}, \breve{\boldsymbol{p}}_u^{sim_1}, \cdots, \breve{\boldsymbol{p}}_u^{sim_q}\right) : u \in \boldsymbol{\mathcal{U}}^{giv}\right\}$, hyperparameters, randomly initialized
        $\mathbf{W}^0, \boldsymbol{a}^0, \boldsymbol{b}^0, \boldsymbol{\alpha}^0, \boldsymbol{\beta}^0$
Output  $\widehat{\mathbf{W}}^0, \widehat{\boldsymbol{a}}^0, \widehat{\boldsymbol{b}}^0, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}$
step 1  Repeat until convergence
step 2    Shuffle $\boldsymbol{\Theta}^{cur}$
step 3    For each $\left(\breve{\boldsymbol{p}}_u^{giv}, \breve{\boldsymbol{p}}_u^{sim_1}, \cdots, \breve{\boldsymbol{p}}_u^{sim_q}\right)$ in $\boldsymbol{\Theta}^{giv}$ :
step 4      $\mathbf{s}^{vis} = \breve{\boldsymbol{p}}_u^{giv}$,
            $\mathbf{t}^{sim_n} = \breve{\boldsymbol{p}}_u^{sim_n}, n = 1,2,\dots,q$;
step 5      For $m = 0,1 \dots, \mathscr{k} - 1$ :
step 6        Sampling $\mathbb{H}_j, j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$: Obtain the state value $h_j^{(m)}$ according to Eq.(2).
Step 7        Reconstructing $\mathbb{V}_i, i = 1,2,\dots,\mathcal{N}_{\mathbb{V}}$: Obtain the state value $s_i^{(m+1)}$ according to Eq.(3).
Step 8      For $j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$ :
              Sampling $\mathbb{H}_j, j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$: Obtain the state value $h_j^{(\mathscr{k})}$ according to Eq.(2).
Step 9      For $i = 1,2,\dots,\mathcal{N}_{\mathbb{V}}, j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$ :
              According to Eq.(A9)~(A11), update the parameters $\widehat{w}_{ij}, \widehat{a}_i, \widehat{b}_j$。
              According to Eq.(A17), update the parameters $\widehat{\alpha}_{ij}^n, \widehat{\beta}_{ii}^n$, $n = 1,2,\dots,q$.
```

D.2 Graph Embedding in V Module

We construct a POI-word heterogeneous network with POIs and POI tag words as nodes, where the set of POIs is $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{V}}^{loc} \cup \boldsymbol{\mathcal{V}}^{out}$. Through the semantic description of POIs and the semantic similarity between words, similar POIs in different cities can be closely related. We use Node2vec [21] to obtain the embedding vectors of POIs and words, then match the most similar POIs across cities according to the cosine similarity of the vectors, and transfer the hometown check-in records of out-of-town visitors.

Recently, some advanced graph embedding technologies have emerged for homogeneous networks, such as LINE [50] and Node2vec [21]. A graph embedding technology of homogeneous networks may not be suitable for heterogeneous networks, mainly because the weights of different types of edges in heterogeneous networks are often not comparable, such as large differences in magnitude [51]. The POI-word network we build is not affected by the difference in edge weights, and both POI-word edge and word-word edge weights use binary values (1 or 0). Therefore, Node2vec can be used for embedding learning of POI-word network. Node2vec has horizontal and vertical search capabilities and can capture the semantic similarity and structural correlation information of network nodes, so that the embedding vector of POIs described by the same words can be as much as possible similar and embedding vectors of words with similar semantics are also as similar as possible. The blue line and its connected nodes in Fig.1(c) represent the random walk path generated by Node2vec. For details of Node2vec technology, please refer to literature [21], whose author open sourced the Node2vec code[2].

[2] https://github.com/aditya-grover/node2vec

D.3 Calculation of TF-IDF Module

If user u has checked in POI v, p_{uv} is a non-zero real number, usually 1 or the number of check-ins. Guo et al. [7] believed that simply setting p_{uv} to 1 or the number of check-ins would limit the model's recommendation accuracy. They designed the Term Frequency-Inverse Document Frequency (TF-IDF) module to calculate p_{uv}. The purpose of this is to provide a weighting strategy that assign a weight (real number) for each word based on its occurrence frequency and its rarity in the entire corpus (check-in history). The idea is borrowed from information retrieval. TF-IDF has been found to outperform simple binary one-hot encoding [7]. During the training process of the model, if user u has not checked in POI v, then p_{uv}=nan, not set to 0. In the prediction process of the model, if user u has not checked in POI v, p_{uv} is input to the model with a value of 0, and the model outputs a non-zero preference value.

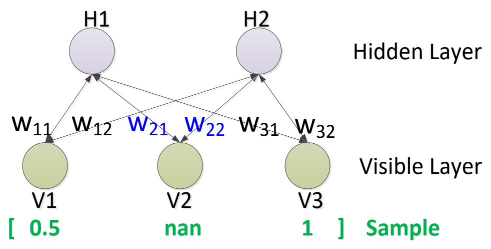


Fig.8 A simple example of RBM receiving missing values during training.

We emphasize that **during model training**, the user's preference value p_{uv} for an unchecked POI is set to the missing value "nan" instead of the "0" value. Fig.8 shows a simple example of RBM receiving missing values during training. When the visible unit V2 receives the missing value "nan", V2 is not reconstructed, and all the connection weights (w21 and w22) between V2 and the hidden layer are frozen. That is, w21 and w22 are not updated to keep the values after the previous sample training. If the missing values are replaced with 0, V2 will be reconstructed with 0 as the state value, which will make the network difficult to converge. This technology was proposed by Hinton et al. [48,52]. It is the key technology to achieve stable training and faster convergence of a RBM network for processing sparse data. We think this is the hardest point of a RBM network. Therefore, in the following involving missing values, we will re-emphasize the specific operation methods to help readers better understand and reproduce the research in this paper.

User $\mathcal{u}$'s preference for the POI $\mathcal{v}$ he/she checked in is [7]:

$$\mathcal{p}_{\mathcal{u}\mathcal{v}} = \mathcal{G}_{\mathcal{u}}^{c} \frac{\mathcal{N}_{\mathcal{u}\mathcal{v}}}{\sum_{z=1}^{|\mathcal{V}_{\mathcal{u}}|} \mathcal{N}_{\mathcal{u}z}} \log(\mathcal{N}_{\mathcal{u}\mathcal{v}} + 1), \tag{A18}$$

where $\mathcal{N}_{\mathcal{u}\mathcal{v}}$ is the number of check-ins of $\mathcal{u}$ to $\mathcal{v}$. $\mathcal{G}_{\mathcal{u}}^{c}$ is the influence weight of POI category $c, c \in \boldsymbol{\mathcal{C}}^{loc}$ on user $\mathcal{u}$:

$$\mathcal{G}_{\mathcal{u}}^{c} = \frac{|\boldsymbol{\mathcal{V}}_{\mathcal{u}}^{c}|}{\sum_{\sigma \in \mathcal{C}_{\mathcal{u}}} |\mathcal{V}_{\mathcal{u}}^{\sigma}|} \log \frac{|\boldsymbol{\mathcal{U}}|+1}{|\boldsymbol{\mathcal{U}}^{c}|}, \tag{A19}$$

where $\boldsymbol{\mathcal{V}}_{\mathcal{u}}^{c}$ represents the POI set that $\mathcal{u}$ has checked in in category c, and $\boldsymbol{\mathcal{U}}^{c}$ represents the set of users who have checked in category c. Note that if $\mathcal{u}$ is an out-of-town visitor, then Eq.(A18) and Eq.(A19) are fed to the number of pseudo check-ins and real check-ins of $\mathcal{u}$ to $\mathcal{v}$.

D.4 TG Supplement

D.4.1 SemiDAE Geographic Impact Model

The SemiDAE geographic influence model [7] is a DL network composed of multiple (Semi)RBM stacks. SemiRBM [48] is a shallow neural network that contains only one visible layer and one hidden layer. The visible layer and the hidden layer are fully connected, and there are full/partial connections between the visible units. The structure of SemiRBM is very suitable for modeling the geographic influence between POIs. It maps visible units to POIs one2one and uses the weighted connection between visible units to characterize geographic similarity. To learn the high-order nonlinear relationship of user-POI interaction, stack multiple RBMs behind SemiRBM to construct a multi-layer SemiDAE. For the theoretical introduction of (Semi)RBM, please refer to Appendix D.1.

Guo et al. [7] analyzed in detail the rationality and advantages of the SemiDAE geographic impact model. SemiDAE uses a network of stacked RBMs to extract high-level features of the input data, which are implicitly stored in each layer of the network. For example, the first hidden layer may represent more detailed POI features such as western food, convenience stores, KTVs, and bookstores, and the higher hidden layer may represent more abstract POI categories such as entertainment, food, and culture. In their experiments, they further proved that the SemiDAE geographic impact model can significantly improve the accuracy of POI recommendation, and it is currently one of the most advanced geographic impact models in the POI recommendation field.

The structure of SemiDAE geographic influence model can refer to Encoder and Decoder in Fig.1(d). In our TGVx model, the specific description of establishing the SemiDAE model for a given time slot $\mathcal{T}^{giv}$ is as follows:

1) The input layer of SemiDAE is the visible layer of SemiRBM#0, the visible unit $\mathbb{V}_i$ corresponds to the POI $\mathcal{v}_i, \mathcal{v}_i \in \boldsymbol{\mathcal{V}}^{loc}$, and the number of visible units is equal to $|\boldsymbol{\mathcal{V}}^{loc}|$. The weighted connection between input (i.e., visible) units characterizes the geographical similarity between POIs. The input layer receives the POI preference vector $\boldsymbol{\mathcal{p}}_u^{giv}$ of the

user $\mathcal{u}$. Because it is impossible for any user to check in all POIs, $\mathcal{p}_u^{giv}$ will have many missing values "nan". When a visible unit receives a missing value, the connection weights between the visible unit and the hidden layer are frozen, and only the visible units that receive the non-missing value are reconstructed [48,52]. Different users have different POI check-in sets, corresponding to different visible unit sets. Even if a visible unit is not trained by the sample of user $\mathcal{u}$, it will be trained by samples of other users' [7].

2) SemiDAE stacks $2R$ (Semi)RBMs: the hidden layer of the previous (Semi)RBM is used as the visible layer of the next RBM. The visible layer and hidden layer of any (Semi)RBM are fully connected. The fully connected weight matrices of SemiRBM#0, RBM#1~RBM#2R-2 and SemiRBM#2R-1 are $\mathbf{W}^0, \mathbf{W}^1, \mathbf{W}^{2R-2}$, and $\mathbf{W}^{2R-1}$.

3) The output layer of SemiDAE is the hidden layer of SemiRBM#2R-1, and the output unit has a one2one correspondence with $|\mathcal{V}^{loc}|$ POIs. The weight connection between output units also represents the geographical similarity between POIs. The output layer of SemiDAE predicts $\mathcal{u}$'s preferences for all POIs on $\mathcal{T}^{giv}$. All POIs are sorted in descending order of preferences, and top-k POIs are recommended to $\mathcal{u}$.

For real-valued POI preference prediction, SemiRBM#0 is a visible unit with real-valued Gaussian, see Appendix D.1.3 for details. For the pre-training algorithm of SemiRBM#0, please refer to Algorithm 2. SemiDAE's RBM#1~RBM#R-2 are RBMs with binary units, see Appendix D.1.1 for details. RBM#R-1 is an RBM with real-valued Gaussian hidden units, see Appendix D.1.2 for details. For the pre-training algorithm of RBM#1~RBM#R-1, please refer to Algorithm 1.

D.4.2 T-SemiRBM#0 Pre-Training Algorithm based on collaborative-social regularization term

The T-SemiRBM#0 pre-training algorithm based on collaborative-social regular items is shown in Algorithm 4, and its process is similar to Algorithm 3. But there are two differences. One is to use the average preference value $\bar{\mathcal{p}}_{\mathcal{F}^{u}i}, i = 1,2,\dots,|\mathcal{V}^{loc}|$ of all users in $\mathcal{F}^{u}$. The other is to get the updated rule Δw_{ij} according to Eq.(4) (step 9).

Algorithm 4 T-SemiRBM#0 pre-training algorithm based on collaborative-social regularization term

```
Input     Training set $\mathbf{\Theta}^{giv} = \left\{\left(\breve{\boldsymbol{\mathcal{p}}}_{u}^{giv}, \breve{\boldsymbol{\mathcal{p}}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{\mathcal{p}}}_{u}^{sim_q}\right) : u \in \boldsymbol{\mathcal{U}}^{giv}\right\}$, hyperparameters; randomly initialized
          $\mathbf{W}^0, \boldsymbol{a}^0, \boldsymbol{b}^0, \boldsymbol{\alpha}^0, \boldsymbol{\beta}^0$; $\bar{\mathcal{p}}_{\mathcal{F}^{u}i}, i = 1,2,\dots,|\mathcal{V}^{loc}|$
Output    $\widehat{\mathbf{W}}^0, \widehat{\boldsymbol{a}}^0, \widehat{\boldsymbol{b}}^0, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}$。
Step 1    **Repeat** until convergence
step 2      Shuffle $\mathbf{\Theta}^{giv}$
step 3      **For each** $\left(\breve{\boldsymbol{\mathcal{p}}}_{u}^{giv}, \breve{\boldsymbol{\mathcal{p}}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{\mathcal{p}}}_{u}^{sim_q}\right)$ **in** $\mathbf{\Theta}^{giv}$ :
step 4        $\mathbf{s}^{vis} = \breve{\boldsymbol{\mathcal{p}}}_{u}^{giv}$,  $\mathbf{t}^{sim_n} = \breve{\boldsymbol{\mathcal{p}}}_{u}^{sim_n}, n = 1,2,\dots,q$;
step 5        **For** $m = 0,1 \dots, \mathcal{k} - 1$ :
step 6          Sampling $\mathbb{H}_j, j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$: Obtain the state value $h_j^{(m)}$ according to Eq.(2).
Step 7          Reconstructing $\mathbb{V}_i, i = 1,2,\dots,\mathcal{N}_{\mathbb{V}}$: Obtain the state value $s_i^{(m+1)}$ according to Eq.(3).
Step 8        **For** $j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$ :
                Sampling $\mathbb{H}_j$: Obtain the state value $h_j^{(\mathcal{k})}$ according to Eq.(2).
Step 9        **For** $i = 1,2,\dots,\mathcal{N}_{\mathbb{V}}, j = 1,2,\dots,\mathcal{N}_{\mathbb{H}}$ :
                According to Eq.(4), update the parameters $\Delta w_{ij}$;
                According to Eq.(A10)~(A11), update the parameters $\hat{a}_i, \hat{b}_j$;
                According to Eq.(A17), update the parameters $\hat{\alpha}_{ij}^{n}, \hat{\beta}_{ii}^{n}$, $n = 1,2,\dots,q$.
```

Research [4] found that the public preferences of all out-of-town users in the target city have a great impact on the POI preferences of an individual out-of-town user. Therefore, for out-of-town users, we also considered the public preference information of all visitors in the target city and integrated it into the training process of T-SemiDAE in the form of regular items. In other words, if u is an out-of-town visitor, $\mathcal{F}^{u}$ includes all visitors in the target city, u's social friends and collaborative users in the target city.

D.4.3 T-SemiDAE Offline Training

Only using the BP algorithm to train T-SemiDAE is easy to make the model fall into the local minimum problem. To solve this issue, we used the two-stage training method proposed by Hinton et al. [53]: pre-training and parameter fine-tuning. The purpose of pre-training is to limit the weights and biases of T-SemiDAE to a certain parameter space, to avoid reducing the learning quality due to random initialization of parameters.

For ease of description, we define the relevant parameters from the overall level of T-SemiDAE, as shown in Fig.1(d). $\mathbf{W}^0$ is the weight matrix connecting the input layer and the first hidden layer $\mathbb{H}^1$. $\mathbf{W}^r$ is the weight matrix connecting the hidden layer $\mathbb{H}^r$ and $\mathbb{H}^{r+1}$, $r = 1,2,\cdots,2R-2$. $\mathbf{W}^{2R-1}$ is the weight matrix connecting the last hidden layer $\mathbb{H}^{2R-1}$ and the output layer. $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^q\}$ is the set of weight matrices connecting the time-conditional network and $\mathbb{H}^1$. $\boldsymbol{\beta} = \{\boldsymbol{\beta}^1, \cdots, \boldsymbol{\beta}^q\}$ is the set of weight vectors connecting the time-conditional network and the output layer. $\mathbf{b}^r$ is the bias vector of $\mathbb{H}^r$, $r = 1,2,\cdots,2R-1$. $\mathbf{b}^{2R}$ is the bias vector of the output layer. The set of all parameters of T-SemiDAE is $\{\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{B}\}$, where $\mathbf{W} = \{\mathbf{W}^0, \dots, \mathbf{W}^{2R-1}\}$, $\mathbf{B} = \{\mathbf{b}^1, \dots, \mathbf{b}^{2R}\}$. The number of hidden units in the encoder decreases layer by layer.

D.4.3.1 Pre-training T-SemiDAE

Because the encoder and decoder of T-SemiDAE are mirror-symmetrical with the middle-hidden layer $\mathbb{H}^R$. The hidden layer $\mathbb{H}^R$ is both the last layer of the encoder and the first layer of the decoder. Therefore, we first pre-train the encoder, and then copy the encoder in a mirror-symmetric manner to obtain a pre-trained decoder. This method greatly improves the training speed of the model [7]. Pre-train T-SemiRBM#0 and RBM#1~RBM#R-1 in the encoder one by one, the specific process is as follows:

First, T-SemiRBM#0 is pre-trained according to Algorithm 4. After pre-training, the weight matrix of T-SemiRBM#0 and its transposed matrix are used as the initial values of $\mathbf{W}^0$ and $\mathbf{W}^{2R-1}$, respectively. The bias vector of the visible layer of T-SemiRBM#0 is used as the initial value of bias vector $\mathbf{b}^{2R}$ of T-SemiDAE. The bias vector of the hidden layer of T-SemiRBM#0 is used as the initial value of the bias vector $\mathbf{b}^1$ of the T-SemiDAE. The connection weight vector between the time-conditional layer $\mathbb{T}^n, n = 1,2,\cdots,q$ and the visible layer of T-SemiRBM#0 is used as the initial value of the connection weight vector $\boldsymbol{\beta}^n$ between $\mathbb{T}^n$ and the T-SemiDAE output layer. The connection weight matrix between $\mathbb{T}^n$ and the hidden layer of T-SemiRBM#0 is used as the initial value of the connection weight matrix $\boldsymbol{\alpha}^n$ between $\mathbb{T}^n$ and the first hidden layer $\mathbb{H}^1$ of T-SemiDAE.

Then, RBM#r,r=1,2...,R-1 is pre-trained according to Algorithm 1. After pre-training, the weight matrix of RBM#r and its transposed matrix are used as the initial values of $\mathbf{W}^r$ and $\mathbf{W}^{2R-1-r}$, respectively. The bias vector of the hidden layer of RBM#r is used as the initial value of the bias vector $\mathbf{b}^{r+1}$ of T-SemiDAE. The bias vector of the visible layer of RBM#r mirrors the initial value of bias vector $\mathbf{b}^{2R-r}$ of T-SemiDAE.

D.4.3.2 Fine-Tuning T-SemiDAE

We use the BP algorithm to fine-tune the parameters of the pre-trained T-SemiDAE. Minimize the overall mean square error loss function between the input layer and output layer of T-SemiDAE:

$$Loss(\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{B}) = \frac{1}{2|\boldsymbol{\mathcal{U}}^{giv}|} \sum_{u \in \boldsymbol{\mathcal{U}}^{giv}} \sum_{i=1,\, s_{ui} \neq nan}^{|\boldsymbol{\mathcal{V}}^{loc}|} \left(\breve{\wp}_{ui}^{giv} - \hat{\wp}_{ui}^{giv}\right)^2 + \frac{\eta}{2}\|\mathbf{W}\|^2, \tag{A20}$$

where η $(\eta > 0)$, is the regularization term coefficient, and $\breve{\mathscr{p}}_{ui}^{giv} \in \breve{\boldsymbol{\mathscr{p}}}_{u}^{giv}$.

In the fine-tuning process, regular items containing information about collaborative users, social friends, and all foreign visitors are added to the update rules:

$$\begin{cases} \Delta w_{ij}^{0} = \Delta w_{ij\_BP}^{0} + \lambda \bar{\mathscr{p}}_{\boldsymbol{\mathcal{F}}^{u}i} h_{j}^{1} \\ \Delta w_{ji}^{2R-1} = \Delta w_{ji\_BP}^{2R-1} + \lambda \bar{\mathscr{p}}_{\boldsymbol{\mathcal{F}}^{u}i} h_{i}^{2R-1}, \end{cases} \tag{A21}$$

where $\Delta w_{ij\_BP}^{0}$ and$\Delta w_{ji_BP}^{2R-1}$ are the weight gradients from the BP algorithm. λ and $\bar{\mathscr{p}}_{\boldsymbol{\mathcal{F}}^{u}i}$ are consistent with those in Eq.(4).

Algorithm 5 The fine-tuning algorithm of T-SemiDAE

```
Input   Training set $\boldsymbol{\Theta}^{giv} = \left\{\left(\breve{\boldsymbol{\mathscr{p}}}_{u}^{giv}, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_q}\right) : u \in \boldsymbol{\mathcal{U}}^{giv}\right\}$, Pre-trained model of T-SemiDAE, $\bar{\mathscr{p}}_{i}^{\mathbb{F}^{u}}, i = 1,2,\dots,|\boldsymbol{\mathcal{V}}|$. $\bar{\mathscr{p}}_{\boldsymbol{\mathcal{F}}^{u}i}, i = 1,2,\dots,|\boldsymbol{\mathcal{V}}^{loc}|$
Output  Optimized $\widehat{\mathbf{W}}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\mathbf{B}}$
step 1  Repeat until convergence
step 2      Shuffle $\boldsymbol{\Theta}^{giv}$
step 3      For each $\left(\breve{\boldsymbol{\mathscr{p}}}_{u}^{giv}, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_q}\right)$ in $\boldsymbol{\Theta}^{giv}$:
step 4          $\mathbf{s}^{vis} = \breve{\boldsymbol{\mathscr{p}}}_{u}^{giv}$, $\mathbf{t}^{sim_n} = \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_n}, n = 1,2,\dots,q$;
step 5          Calculate the output of $\mathbb{H}^{1}$ according to Eq.(A22);
                Calculate the output of $\mathbb{H}^{r} (r = 2,\cdots,2R-1, r \neq R)$ according to Eq.(A23);
                Calculate the output of $\mathbb{H}^{R}$ according to Eq.(A24);
                Calculate $\widehat{\boldsymbol{\mathscr{p}}}_{u}^{giv}$ according to Eq.(A25).
Step 6          Based on BP algorithm and Eq.(A20) update parameters $\widehat{\mathbf{W}}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\mathbf{B}}$;
step 7          According to Eq.(A21), the update rules $\Delta w_{ij}^{0}$ and $\Delta w_{ji}^{2R-1}$ are obtained, and the parameters $\widehat{\mathrm{W}}^{0}$ and $\widehat{\mathrm{W}}^{2R-1}$ are further optimized.
```

Refer to Algorithm 5 for the fine-tuning algorithm of T-SemiDAE. The T-SemiDAE pre-training model for a given time slot, the training set $\boldsymbol{\Theta}^{giv} = \left\{\left(\breve{\boldsymbol{\mathscr{p}}}_{u}^{giv}, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{\mathscr{p}}}_{u}^{sim_q}\right) : u \in \boldsymbol{\mathcal{U}}^{giv}\right\}$, the average preference value $\bar{\mathscr{p}}_{\boldsymbol{\mathcal{F}}^{u}i}, i = 1,2,\dots,|\boldsymbol{\mathcal{V}}^{loc}|$ of all users in $\boldsymbol{\mathcal{F}}^{u}$ are known. Each time a user sample is extracted from $\boldsymbol{\Theta}^{giv}$ and fed to the input layer and the time-conditional network (steps 3~4). Calculate the predicted value $\widehat{\boldsymbol{\mathscr{p}}}_{u}^{giv}$ of the output layer according to Eq.(A22)~(A25) (step 5). Then, the parameters are fine-tuned, and the parameters $\widehat{\mathbf{W}}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\mathbf{B}}$ are updated based on the BP algorithm and Eq.(A20) (step 6). According to Eq.(A21), the update rules Δw_{ij}^{0} and Δw_{ji}^{2R-1} are obtained, and the parameters $\widehat{\mathbf{W}}^{0}$ and $\widehat{\mathbf{W}}^{2R-1}$ are further optimized (step 7). Repeat training for multiple rounds as needed until convergence (steps 1-7), and shuffle $\boldsymbol{\Theta}^{cur}$ (step 2) before each round of training. Finally, the optimized $\widehat{\mathbf{W}}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\mathbf{B}}$ are obtained. When an input unit receives a missing value, freeze the connection weights of that unit to the first hidden layer. When a temporal conditional network unit receives a missing value, the connection weights of the unit to the first hidden layer and the output layer are frozen.

D.4.4 T-SemiDAE Online Prediction

The input layer receives $\breve{\boldsymbol{p}}_{u}^{giv}$, and the time-conditional network receives $\breve{\boldsymbol{p}}_{u}^{sim_1}, \cdots, \breve{\boldsymbol{p}}_{u}^{sim_q}$. The missing value is set to 0, and the non-zero preference value is the output after being predicted by the T-SemiDAE model.

For $\mathbb{H}^1$, the output value of the hidden unit $\mathbb{H}_j^1$ is:

$$y_j^1 = ReLU\left(b_j^1 + \sum_{i=1}^{|\boldsymbol{\mathcal{V}}^{loc}|} s_i w_{ij}^0 + \sum_{n=1}^{q} \sum_{i=1}^{|\boldsymbol{\mathcal{V}}^{loc}|} t_i^{sim_n} \alpha_{ij}^n\right), \tag{A22}$$

where $b_j^1 \in \mathbf{b}^1$.

For $\mathbb{H}^r (r = 2, \cdots, 2R-1, r \neq R)$, the output value of the hidden unit $\mathbb{H}_j^r$ is:

$$y_j^r = ReLU\left(b_j^r + \sum_{z=1} h_z^{r-1} w_{zj}^{r-1}\right), \tag{A23}$$

where $b_j^r \in \mathbf{b}^r$.

For $\mathbb{H}^R$, the output value of the hidden unit $\mathbb{H}_j^R$ is sampled from the normal distribution:

$$y_j^R = \mathbb{N}\left(b_j^R + \sum_{z=1} h_z^{R-1} w_{zj}^{R-1}, 1\right), \tag{A24}$$

The prediction preference of the output unit $\mathbb{O}_i, i = 1,2, \cdots, |\boldsymbol{\mathcal{V}}^{loc}|$ is:

$$\hat{\mathscr{p}}_{ui}^{giv} = b_i^{2R} + \sum_{j=1} h_j^{2R-1} w_{ji}^{2R-1} + \sum_{z=1,z\neq i}^{|\boldsymbol{\mathcal{V}}^{loc}|} \gamma_{iz} s_z + \sum_{n=1}^{q} t_i^{sim_n} \beta_{ii}^{n}. \tag{A25}$$

Appendix E Experiments Supplement

E.1 Other Experiment Setup

E.1.1 Example of the out-of-town area of a target city

For example, the out-of-town area of Tokyo was a Japanese area centered on Tokyo's geographic coordinates and a radius greater than 100km. The method in [16] was used to distinguish between local and out-of-town users in the target city. For basic information of the target city dataset, please see Table 1.

E.1.2 Detailed Experiment Design

The training set and the validation set are respectively used for model learning and hyperparameter selection, and the test set simulates the online application process. The check-in activities of the training set and the validation set occur before the check-in activities of the test set, which was in line with the actual situation. The recommended accuracy of the TG(V)x model was evaluated on all subtest sets, and the ablation experiments as well as the "TG(V)x vs. other advanced models" experiments were performed. The recommended diversity of the TG(V)x model was evaluated on the warm-start and out-of-town test sets, and the experiment of "TG(V)x vs. other advanced models" was carried out. We also performed another two experiments "nonlinear design Vs. linear design" and "unequal time slots vs. equal time slots".

Previous studies have mostly focused on recommending novel POIs for users, which is certainly important. However, it is practical to recommend POIs that have been visited by users, because this is in line with the phenomenon of “returning customers” retained by businesses and the competitive relationship between similar POIs, and it can also remind users to visit POIs that have not been visited for a long time. The TGVx model proposed in this paper adopts a "one2all" output mode, which can flexibly recommend including or not including visited POIs according to the needs of the recommender system. Considering the phenomenon of "returning customers" and the competition of similar POIs, the test set of this paper includes the POIs that users have visited. In addition, cold-start users and out-of-town visitors have very few check-in records in the target city. For these two groups, whether the POIs visited in the test set is included has little effect on the evaluation results of the model. For warm-start users, if the test set contains visited POIs, the model may be more inclined to recommend the visited POIs, which may improve accuracy while reducing the diversity of recommendations. A good POI recommendation model should consider accuracy and diversity at the same time. Even under the condition that the visited POIs are included in the test set, the accuracy and diversity can be improved at the same time.

E.1.3 Detailed Baseline Models

In both local and out-of-town scenarios, we compared the performance of the TG(V)x model with the following advanced models:

- **Semi-CDAE** [7]: A static POI deep recommendation model proposed by Guo et al. Semi-CDAE adopts TF-IDF technology to transform POI semantic information, uses SemiRBM visible layer to represent geographic information, receives social influence with conditional network, and realizes the effective integration of geographic, social and semantic information. It fully exploits high-order non-linear user-POI interactions through DAE stacked by (Semi)RBM, and significantly improves the accuracy of POI recommendations.

- **LBSN2vec** [13]: A POI recommendation model based on hypergraph embedding proposed Yang et al. It samples user check-in information and social relationships from a heterogeneous network graph composed of LBSN data, then learns node embedding, and finally obtains the POI recommendation ranking by calculating the similarity of node embedding vectors. LBSN2vec can effectively integrate time, geographic, semantic, and social information, and provide POI recommendations with time dynamics.
- **LFBCA** [15]: A POI recommendation model based on graph embedding proposed by Wang et al. The model creates POI recommendations based on geographic, collaborative, and social information.
- **USG** [14]: A collaborative filtering recommendation model based on Naive Bayes proposed by Ye et al. It integrates geographic, collaborative, and social information into POI recommendations.
- **LRT** [10]: A POI recommendation model based on time-enhanced MF proposed by Gao et al. The LRT framework includes three steps: time division, time factorization, and time aggregation, and a time regularization term is added to the objective function to model time continuity. LRT is a linear shallow model that only uses simple summation, averaging, voting and other methods to achieve time aggregation.

Table 3 summarizes the influencing factors and basic methods considered by TGVx and other advanced models. These models were coded in Python and Matlab and implemented in Windows 10. When recommending for user u, TG(V)x and all baseline models predicted u's preference for all POIs in the target city and took top-k to generate a recommendation list.

Table 3 Influencing factors and basic methods considered by TGVx and other advanced models

Models	Factors / Basic Methods	Spatial	Temporal	Semantic	Social	Collaborative	Public
TGx	DL	●	●	●	●	●	
TGVx	DL	●	●	●	●	●	●
Semi-CDAE	DL	●		●	●		
LBSN2Vec	Hypergraph	●	●	●	●		
LFBCA	Graph	●			●	●	
USG	Bayesian-CF	●			●	●	
LRT	Time-MF		●				

E.1.4 Detailed Evaluation Metric

When evaluating the performance of POI recommendation models, previous work mostly considered the accuracy of recommendations. Recall rate refers to the proportion of restored POIs to visited POIs. NDCG measures ranking quality, and the higher the ranking, the higher the score of POIs. We did not use precision as an evaluation metric because precision is not suitable for measuring recommendation models based on implicit feedback [29,36].

The top-k recommendation list generated for user u is $\mathcal{R}_u, k = \{5,10,15,20\}$. Recall@$k$ is defined as:

$$\text{Racall@}k = \frac{\mathcal{N}_{positive}}{\mathcal{N}_{positive}+\mathcal{N}_{negative}}, \tag{A26}$$

where $\mathcal{N}_{positive}$ represents the number of POIs accessed by u in $\mathcal{R}_u$, and $\mathcal{N}_{negative}$ represents the number of POIs accessed by user u but not in $\mathcal{R}_u$.

NDCG@k is defined as:

$$\mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}, \tag{A27}$$

where

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{2^{cor_i - 1}}{log_2(i+1)}, \tag{A28}$$

where cor_i represents the level correlation of the i-th result. Herer, we use binary correlation: if the result is in $\mathcal{R}_u$, then cor_i=1, otherwise it is 0. IDCG@k is the value of DCG@k when ideally ranking recommended POIs. The value range of Recall@k and NDCG@k are both [0,1], the larger the value, the higher the accuracy. The average value of all users in any test set is used as the final Recall@k and NDCG@k.

Diversity is the soul of personalized recommendations, and it is an effective guarantee for recommending different POIs for different users. According to $\mathcal{R}_u$'s Intra-List similarity $\frac{1}{k(k-1)}\sum_{i\neq j} s_{ij}$ [24,25], we define $\mathcal{R}_u$'s Intra-List diversity as:

$$\mathrm{Intra@}k = 1 - \frac{1}{k(k-1)}\sum_{i\neq j} s_{ij}, \tag{A29}$$

where s_{ij} is the similarity between POI v_i and v_j in $\mathcal{R}_u$ and $s_{ij} \in [0,1]$. Here we use the similarity of POIs embedding vectors, see Section 4.2 for details. The value range of Intra@k is [0,1]. The larger the value, the higher the individual diversity of $\mathcal{R}_u$. The average value of all users in any test set is used as the Intra@k of a recommended model.

The recommended lists of users u and u' are $\mathcal{R}_u$ and $\mathcal{R}_{u'}$, respectively. By quantifying the distance $1 - |\mathcal{R}_u \cap \mathcal{R}_{u'}|/k$ between $\mathcal{R}_u$ and $\mathcal{R}_{u'}$, the Inter-List diversity is defined as [25]:

$$\mathrm{Inter@}k = \frac{1}{|\mathcal{U}_{test}|(|\mathcal{U}_{test}|-1)}\sum_{i\neq j} 1 - \frac{|\mathcal{R}_u \cap \mathcal{R}_{u'}|}{k}, \tag{A30}$$

where $|\mathcal{U}_{test}|$ represents the number of users in the test set. The value range of Inter@k is [0,1]. The greater the distance between the recommendation lists, the greater the Inter@k

value, the higher the degree of personalization, and the better the aggregate diversity of recommendations.

We used the average of the metric values (e.g., Recall@k) of x TG(V)s as the metric value of TG(V)x. In the same way, the simplified models of TG(V)x also obtained their metric values.

E.1.5 Detailed Parameters

Calculate the distance d between two POIs based on the latitude and longitude coordinates. The nonlinear least squares method was used to fit the check-in probability with the geographical distance between POIs to obtain the power-law distribution parameters ϖ and ξ. Feed ϖ, ξ, d into Eq.(1) to obtain geographic similarity.

Table 4 The Detailed Parameter Values of the TGVx models

Parameter	Value	Parameter	Value
ϖ(Tokyo, Istanbul, New York)	0.08755, 0.09945,0.11047	dimension of embedding vectors	32
ξ(Tokyo, Istanbul, New York)	-0.94975, -0.88948,-1.20021	step length of random walk	10
learning rate(Tokyo, Istanbul, New York)	0.002, 0.001, 0.002	number of random walks	80
epochs(Tokyo, Istanbul, New York)	50, 80,75	horizontal and vertical parameters	0.25,0.4
batch size	256	q	1
number of units $\mathbb{H}^1 \sim \mathbb{H}^5$	2000, 1000, 500, 1000, 2000	σ	1
x	12	cosine similarity threshold $\emptyset$	0.9

The TGVx models of the three target cities used the same structure. Set the hyperparameter x according to the amount of data. The larger the value of x, the better the time sensitivity of the TGVx model, but the degree of data sparsity is increased. We set x=12, and the time slots of the three target cities after clustering were shown in Fig. 3. For the TGV model in any time slot, its structure was T-SemiRBM#0→RMB#1→RMB#2→RMB#3 →RMB#4→T-SemiRBM#5, and there were 5 hidden layers in total. The number of input (output) units was the total number of POIs in the target city, see Table 1 for details. q was an empirical value given by us based on the data scale. It was also considered that if more similar time slots are used, noise interference may be enhanced, which was not conducive to obtaining more accurate recommendation results.

When using LBSN2Vec, the cosine similarity threshold $\emptyset(0 \leq \emptyset \leq 1)$ (see Appendix B.3) was required to determine the range of potential friends and collaborative users. We found that as the value of $\emptyset$ decreases, the number of potential friends and collaborative users mined would increase, expanding the scope of collaboration and social influence, but at the same time it would also bring more noise. When $\emptyset$=0.9, the result was the best. Because the number of user node pairs was too large, we randomly selected 5% of user node pairs to calculate the cosine similarity of the user nodes embedding vector.

E.2 Results of Ablation Experiments

Ablation experiments are usually performed in two ways. One is the full permutation and combination method [7], which is suitable for models with a small number of functional modules. The other is the leave-one-out method [4], i.e., each simplified model only removes one functional module. Since our TGVx model contains many functional modules that consider both local and non-local recommended scenarios, and it is not easy to sort out the experimental conclusions by using the full permutation and combination method, we choose the leave-one-out method for the ablation study.

For **local recommendations**, TGx's simplified versions are:

- TGx-G: Exclude geographic weights and examine the effectiveness of geographic information in local recommendations.
- TGx-T: Eliminate the time-conditional network and examine the effectiveness of time

information in local recommendations.

- TGx-S: Eliminate social relationships in regular items and examine the effectiveness of social information in local recommendations.
- TGx-C: Eliminate collaborative users in the regularization terms and examine the effectiveness of collaborative information in local recommendations.

For **out-of-town recommendations**, TGVx's simplified versions are:

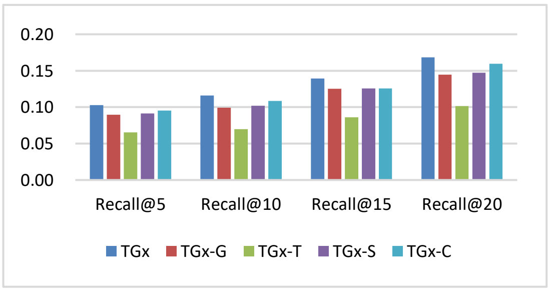
- TGVx-G: Exclude geographic weights and examine the effectiveness of geographic information in out-ot-town recommendations.
- TGVx-T: Eliminate the time-conditional network and examine the effectiveness of time information in out-ot-town recommendations.
- TGVx-S: Excluding the social relationship data from the regular items and examine the effectiveness of social information in out-ot-town recommendations.
- TGVx-C: Eliminate the collaborative users in the regular items and examine the effectiveness of collaborative information in out-ot-town recommendations.

Table 5 Ablation experiments w.r.t. accuracy on Tokyo dataset

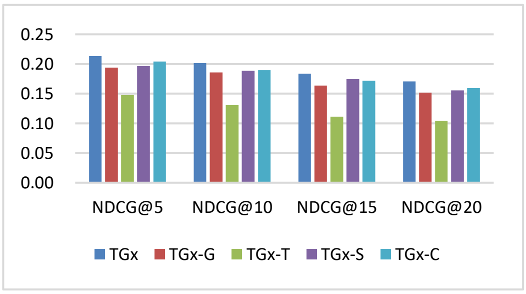
	Recall					**NDCG**				
Model	@5	@10	@15	@20	$\overline{\downarrow @k}\%$	@5	@10	@15	@20	$\overline{\downarrow @k}\%$
warm-start										
TGx	0.103	0.116	0.139	0.168	0.0	0.213	0.201	0.183	0.171	0.0
TGx-G	0.090	0.099	0.125	0.145	-12.9	0.194	0.186	0.163	0.152	-9.7
TGx-T	0.065	0.070	0.086	0.101	-38.6	0.148	0.131	0.111	0.104	-36.0
TGx-S	0.091	0.102	0.126	0.147	-11.4	0.196	0.188	0.175	0.156	-7.1
TGx-C	0.095	0.109	0.126	0.159	-7.3	0.204	0.190	0.172	0.159	-5.9
Out-of-town										
TGVx	0.059	0.071	0.090	0.118	0.0	0.077	0.071	0.063	0.054	0.0
TGVx-G	0.052	0.062	0.081	0.101	-12.6	0.071	0.067	0.060	0.049	-6.6
TGVx-T	0.035	0.041	0.055	0.078	-38.6	0.057	0.047	0.042	0.036	-31.6
TGVx-S	0.055	0.063	0.085	0.110	-7.4	0.073	0.068	0.059	0.051	-5.2
TGVx-C	0.048	0.055	0.075	0.095	-19.1	0.068	0.061	0.056	0.047	-12.4

Table 6 Ablation experiments w.r.t. accuracy on Istanbul Dataset

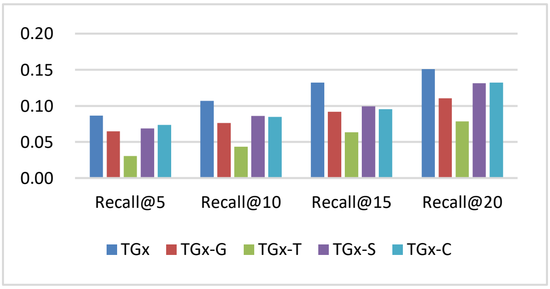
	Recall					NDCG				
Model	@5	@10	@15	@20	$\downarrow \overline{@k}\%$	@5	@10	@15	@20	$\downarrow \overline{@k}\%$
warm-start										
TGx	0.086	0.107	0.132	0.151	0.0	0.180	0.167	0.150	0.136	0.0
TGx-G	0.065	0.076	0.092	0.110	-27.8	0.160	0.149	0.134	0.127	-9.8
TGx-T	0.030	0.043	0.063	0.079	-56.1	0.101	0.088	0.079	0.073	-46.3
TGx-S	0.069	0.086	0.099	0.131	-19.5	0.166	0.149	0.134	0.125	-9.3
TGx-C	0.073	0.085	0.096	0.132	-19.0	0.159	0.145	0.131	0.115	-13.2
Out-of-town										
TGVx	0.044	0.058	0.069	0.094	0.0	0.072	0.062	0.052	0.045	0.0
TGVx-G	0.040	0.049	0.061	0.076	-13.5	0.054	0.044	0.041	0.038	-22.5
TGVx-T	0.025	0.035	0.043	0.056	-40.5	0.044	0.041	0.033	0.030	-35.9
TGVx-S	0.042	0.054	0.068	0.086	-5.7	0.065	0.054	0.046	0.043	-9.8
TGVx-C	0.033	0.048	0.061	0.082	-16.1	0.055	0.048	0.041	0.034	-22.9



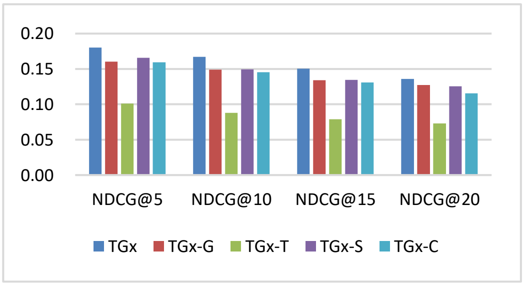
(a) Tokyo: Recall



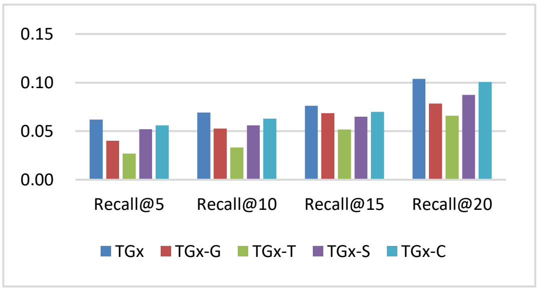
(b) Tokyo: NDCG



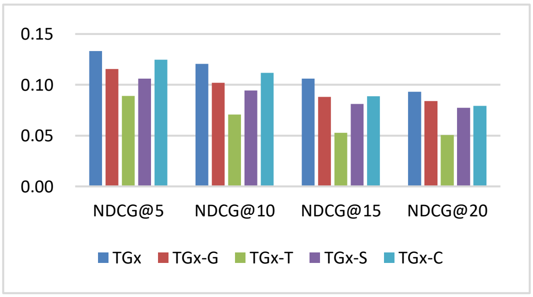
(c) Istanbul: Recall



(d) Istanbul: NDCG



(e) New York: Recall

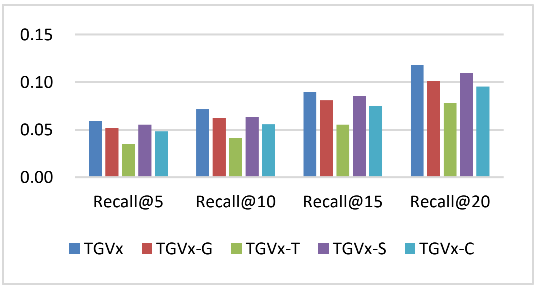


(f) New York: NDCG

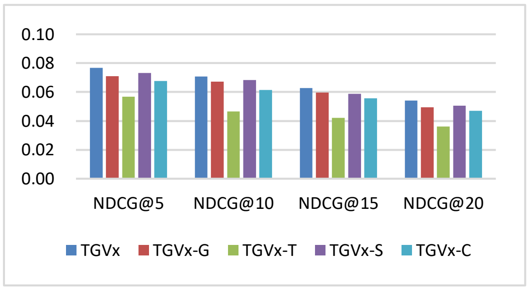
Fig.9 The accuracy of TGx and its simplified models for warm-start users.

Table 7 Ablation experiments w.r.t. accuracy on New York Dataset

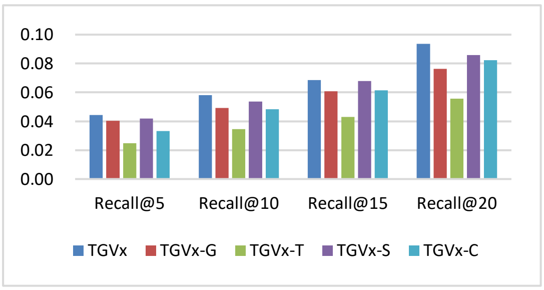
	Recall					NDCG				
Model	@5	@10	@15	@20	↓$\overline{@k}$%	@5	@10	@15	@20	↓$\overline{@k}$%
warm-start										
TGx	0.062	0.069	0.076	0.104	0.0	0.133	0.121	0.106	0.093	0.0
TGx-G	0.040	0.053	0.068	0.079	-23.4	0.116	0.102	0.088	0.084	-13.8
TGx-T	0.027	0.033	0.052	0.066	-44.1	0.089	0.071	0.053	0.050	-42.6
TGx-S	0.052	0.056	0.065	0.087	-16.4	0.106	0.095	0.081	0.077	-20.6
TGx-C	0.056	0.063	0.070	0.101	-7.5	0.125	0.112	0.089	0.079	-11.2
out-of-town										
TGVx	0.037	0.041	0.045	0.061	0.0	0.045	0.040	0.037	0.030	0.0
TGVx-G	0.032	0.037	0.042	0.055	-9.5	0.041	0.036	0.031	0.026	-12.2
TGVx-T	0.017	0.024	0.031	0.045	-37.2	0.026	0.021	0.016	0.014	-50.0
TGVx-S	0.034	0.033	0.041	0.054	-11.7	0.042	0.037	0.030	0.027	-10.3
TGVx-C	0.027	0.031	0.038	0.049	-21.1	0.033	0.028	0.023	0.020	-32.3



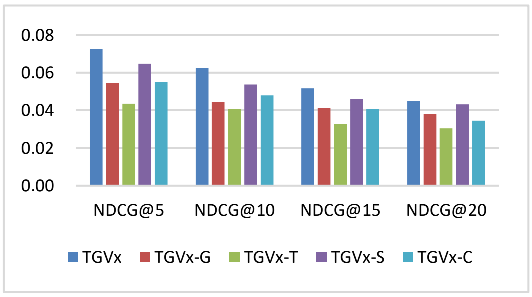
(a) Tokyo: Recall



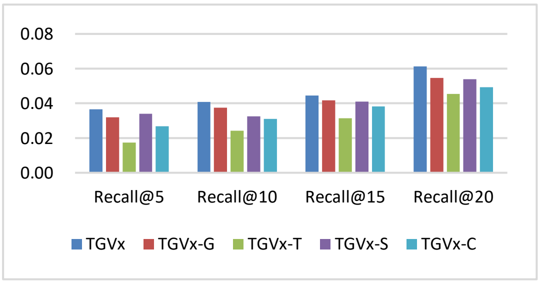
(b) Tokyo: NDCG



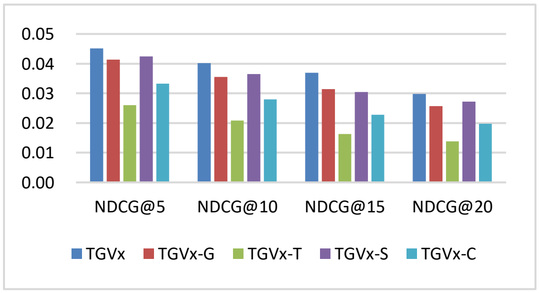
(c) Istanbul: Recall



(d) Istanbul: NDCG



(e) New York: Recall



(f) New York: NDCG

Fig.10 The accuracy of TGVx and its simplified models for out-of-town users.

E.3 Results of the "TG(V)x vs. Other Advanced Models" Experiment

Table 8 “TG(V)x vs. other advanced models” w.r.t. accuracy on Tokyo Dataset

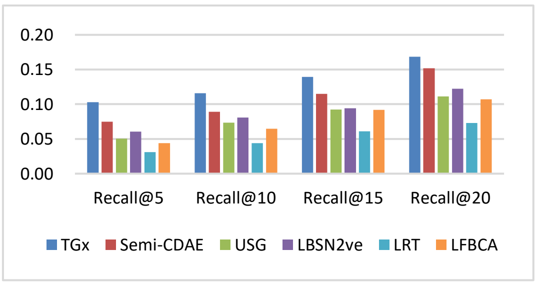
	Recall					NDCG				
Model	@5	@10	@15	@20	$\uparrow \overline{@k}\%$	@5	@10	@15	@20	$\uparrow \overline{@k}\%$
warm-start										
TGx	0.103	0.116	0.139	0.168	163.8	0.213	0.201	0.183	0.171	207.8
Semi-CDAE	0.075	0.089	0.115	0.152	110.4	0.186	0.172	0.160	0.140	163.3
USG	0.051	0.074	0.092	0.111	58.7	0.118	0.113	0.105	0.095	72.5
LBSN2ve	0.061	0.081	0.094	0.122	75.4	0.136	0.118	0.109	0.101	85.3
LRT	0.031	0.044	0.061	0.073	0.0	0.068	0.062	0.060	0.059	0.0
LFBCA	0.044	0.065	0.092	0.107	46.7	0.090	0.086	0.081	0.069	30.7
out-of-town										
TGVx	0.059	0.071	0.090	0.118	130.0	0.077	0.071	0.063	0.054	223.9
Semi-CDAE	0.041	0.047	0.058	0.080	53.9	0.050	0.045	0.041	0.032	104.3
USG	0.027	0.036	0.044	0.055	9.9	0.037	0.030	0.026	0.019	34.0
LBSN2ve	0.035	0.042	0.052	0.074	38.2	0.047	0.041	0.038	0.035	98.3
LRT	0.023	0.033	0.043	0.050	0.0	0.025	0.022	0.019	0.016	0.0
LFBCA	0.027	0.036	0.042	0.060	10.6	0.031	0.029	0.026	0.022	31.0

Table 9 “TG(V)x vs. other advanced models” w.r.t. accuracy on Istanbul Dataset

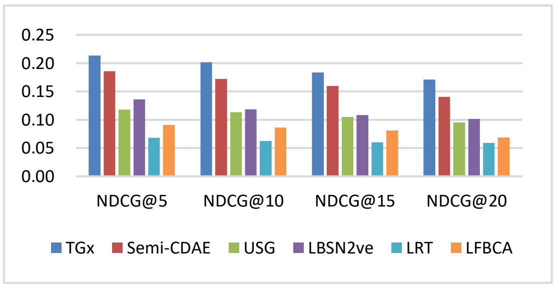
	Recall					NDCG				
Model	@5	@10	@15	@20	$\uparrow \overline{@k}\%$	@5	@10	@15	@20	$\uparrow \overline{@k}\%$
warm-start										
TGx	0.086	0.107	0.132	0.151	358.0	0.180	0.167	0.150	0.136	243.8
Semi-CDAE	0.063	0.074	0.092	0.126	237.0	0.142	0.143	0.123	0.113	182.8
USG	0.030	0.038	0.048	0.060	66.1	0.101	0.092	0.083	0.075	90.6
LBSN2ve	0.052	0.057	0.074	0.105	174.8	0.129	0.116	0.094	0.097	136.0
LRT	0.017	0.022	0.035	0.033	0.0	0.053	0.048	0.044	0.039	0.0
LFBCA	0.026	0.039	0.044	0.059	59.2	0.086	0.080	0.070	0.055	56.8
out-of-town										
TGVx	0.044	0.058	0.069	0.094	223.5	0.072	0.062	0.052	0.045	221.2
Semi-CDAE	0.036	0.043	0.055	0.073	153.9	0.053	0.047	0.041	0.032	139.7
USG	0.026	0.028	0.030	0.041	57.6	0.038	0.029	0.026	0.020	55.9
LBSN2ve	0.028	0.031	0.039	0.054	88.9	0.037	0.031	0.025	0.020	56.2
LRT	0.012	0.017	0.025	0.029	0.0	0.022	0.020	0.017	0.013	0.0
LFBCA	0.017	0.020	0.024	0.029	13.4	0.028	0.023	0.020	0.016	20.1

Table 10 “TG(V)x vs. other advanced models” w.r.t. accuracy on New York Dataset

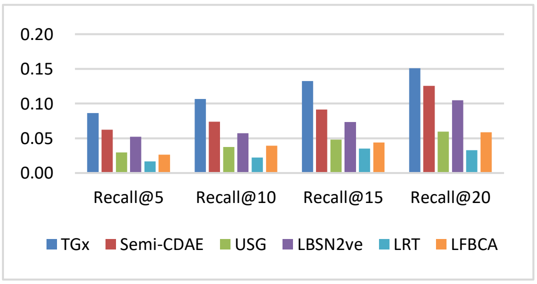
	Recall					**NDCG**				
Model	*@5*	*@10*	*@15*	*@20*	$\overline{\uparrow @k}\%$	*@5*	*@10*	*@15*	*@20*	$\overline{\uparrow @k}\%$
warm-start										
TGx	0.062	0.069	0.076	0.104	195.3	0.133	0.121	0.106	0.093	245.1
Semi-CDAE	0.044	0.049	0.059	0.079	118.4	0.106	0.088	0.071	0.063	146.8
USG	0.023	0.033	0.043	0.057	43.7	0.056	0.050	0.043	0.036	40.1
LBSN2ve	0.036	0.049	0.059	0.077	107.2	0.095	0.080	0.070	0.060	130.5
LRT	0.019	0.022	0.026	0.042	0.0	0.044	0.036	0.030	0.024	0.0
LFBCA	0.022	0.032	0.042	0.054	37.8	0.046	0.042	0.038	0.031	19.3
out-of-town										
TGVx	0.037	0.041	0.045	0.061	341.3	0.045	0.040	0.037	0.030	502.3
Semi-CDAE	0.022	0.025	0.035	0.044	196.2	0.036	0.030	0.026	0.022	345.5
USG	0.017	0.019	0.023	0.028	108.8	0.023	0.019	0.016	0.012	165.5
LBSN2ve	0.018	0.024	0.033	0.042	174.1	0.031	0.025	0.023	0.021	297.6
LRT	0.008	0.009	0.012	0.014	0.0	0.011	0.009	0.006	0.004	0.0
LFBCA	0.015	0.018	0.022	0.028	96.9	0.011	0.009	0.007	0.005	16.8



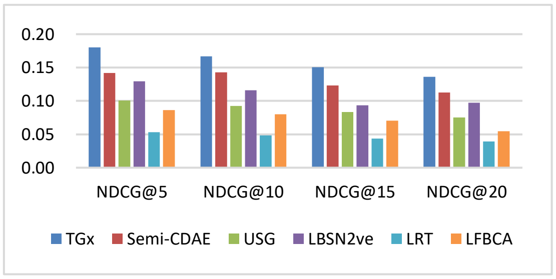
(a) Tokyo: Recall



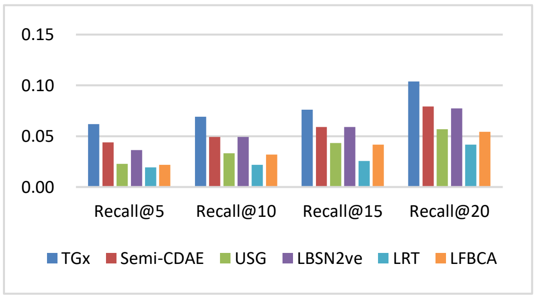
(b) Tokyo: NDCG



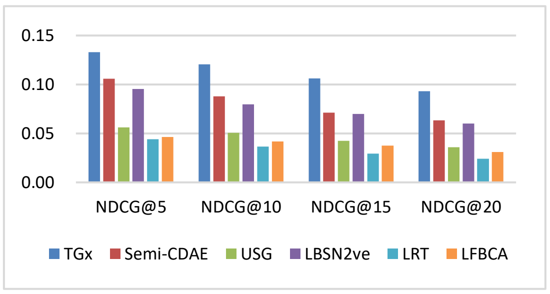
(c) Istanbul: Recall



(d) Istanbul: NDCG

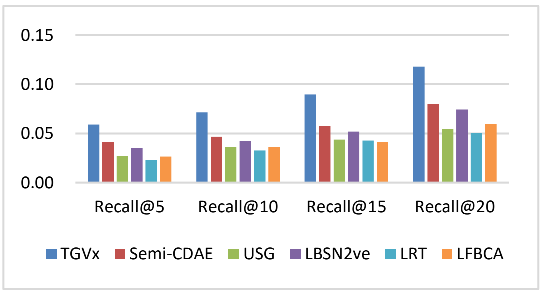


(e) New York: Recall

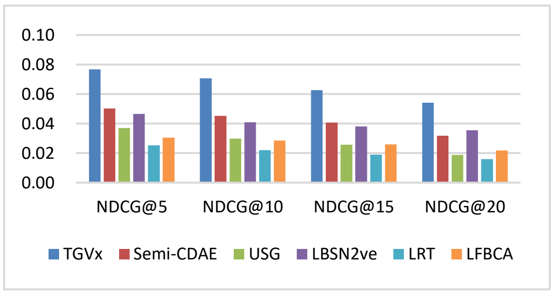


(f) New York: NDCG

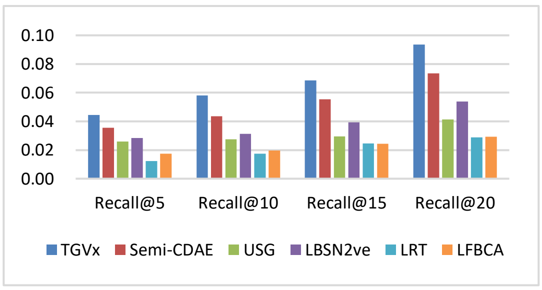
Fig.11 The accuracy of TGx and other advanced models warm-start users.



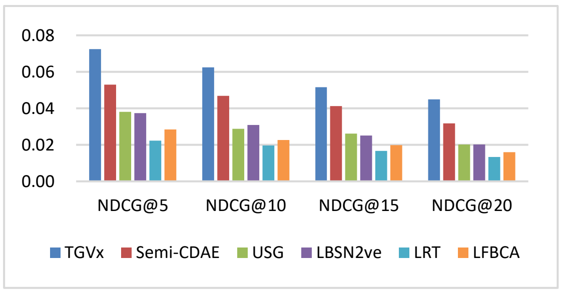
(a) Tokyo: Recall



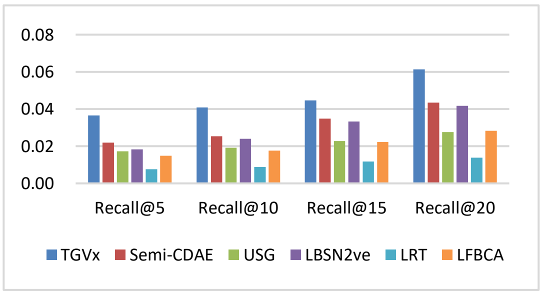
(b) Tokyo: NDCG



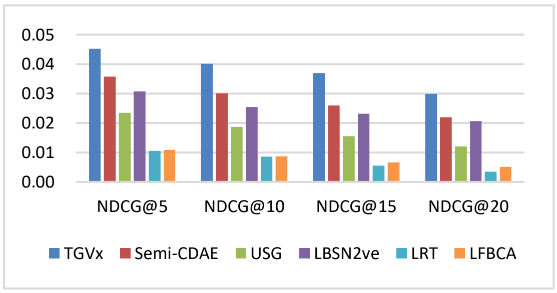
(c) Istanbul: Recall



(d) Istanbul: NDCG



(e) New York: Recall



(f) New York: NDCG

Fig.12 The accuracy of TGVx and other advanced models for out-of-town users.

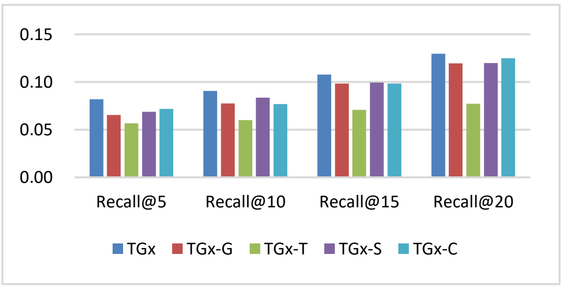
E.4 Results of the Cold-Start Experiments

Table 11 Cold-start experiments w.r.t. accuracy on Tokyo Dataset

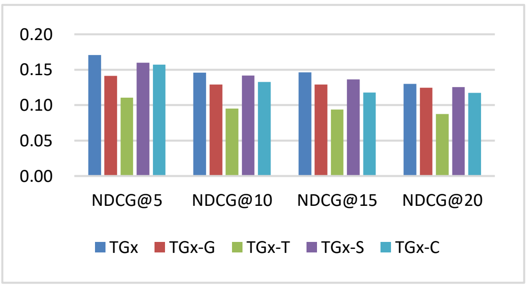
	Recall					NDCG				
Model	@5	@10	@15	@20	$\downarrow \overline{@k}\%$	@5	@10	@15	@20	$\downarrow \overline{@k}\%$
TGx	0.082	0.091	0.108	0.130	0.0	0.171	0.146	0.146	0.130	0.0
TGx-G	0.065	0.077	0.098	0.119	-12.9	0.142	0.129	0.129	0.124	-11.1
TGx-T	0.057	0.060	0.071	0.077	-34.9	0.110	0.095	0.094	0.087	-34.6
TGx-S	0.069	0.083	0.099	0.120	-9.9	0.160	0.142	0.136	0.125	-4.9
TGx-C	0.072	0.077	0.098	0.125	-10.1	0.157	0.133	0.118	0.118	-11.4
Model	@5	@10	@15	@20	$\uparrow \overline{@k}\%$	@5	@10	@15	@20	$\uparrow \overline{@k}\%$
TGx					152.4					224.4
Semi-CDAE	0.057	0.067	0.086	0.104	89.2	0.132	0.113	0.096	0.088	133.1
USG	0.041	0.057	0.074	0.082	52.6	0.111	0.089	0.077	0.073	90.8
LBSN2ve	0.046	0.059	0.076	0.086	60.8	0.109	0.084	0.074	0.067	80.7
LRT	0.027	0.035	0.043	0.066	0.0	0.051	0.047	0.043	0.041	0.0
LFBCA	0.027	0.052	0.069	0.085	35.0	0.069	0.056	0.049	0.043	17.8

Table 12 Cold-start experiments w.r.t. accuracy on Istanbul Dataset

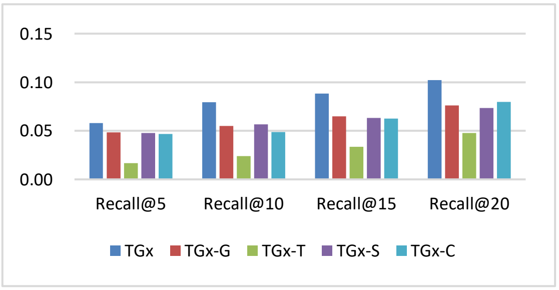
	Recall					NDCG				
Model	@5	@10	@15	@20	↓ $\overline{@k}\%$	@5	@10	@15	@20	↓ $\overline{@k}\%$
TGx	0.058	0.079	0.088	0.102	0.0	0.129	0.104	0.097	0.087	0.0
TGx-G	0.048	0.055	0.065	0.076	-24.8	0.114	0.084	0.077	0.074	-16.3
TGx-T	0.017	0.024	0.034	0.048	-64.1	0.053	0.077	0.051	0.052	-42.9
TGx-S	0.048	0.057	0.063	0.073	-25.6	0.114	0.094	0.079	0.075	-13.1
TGx-C	0.047	0.049	0.062	0.080	-27.4	0.105	0.086	0.077	0.068	-19.3
Model	@5	@10	@15	@20	↑ $\overline{@k}\%$	@5	@10	@15	@20	↑ $\overline{@k}\%$
TGx					311.7					274.4
Semi-CDAE	0.041	0.056	0.062	0.088	204.0	0.083	0.077	0.069	0.062	163.2
USG	0.021	0.034	0.042	0.053	78.2	0.067	0.064	0.052	0.044	103.0
LBSN2ve	0.022	0.032	0.050	0.067	96.0	0.077	0.069	0.052	0.048	118.5
LRT	0.011	0.016	0.028	0.033	0.0	0.037	0.031	0.024	0.021	0.0
LFBCA	0.021	0.026	0.033	0.041	49.5	0.054	0.047	0.042	0.031	55.5



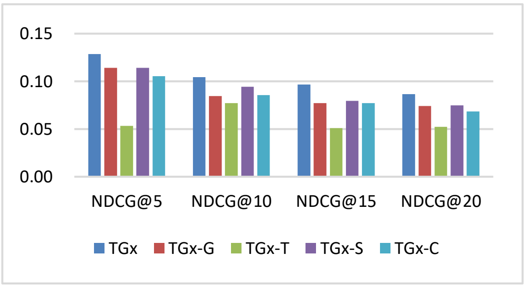
(a) Tokyo: Recall



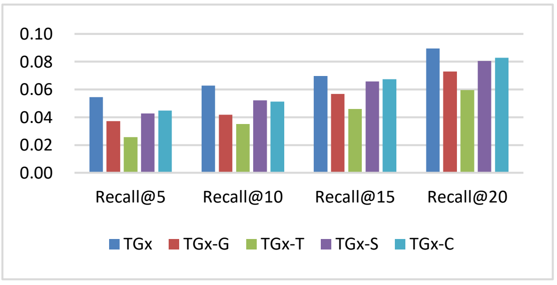
(b) Tokyo: NDCG



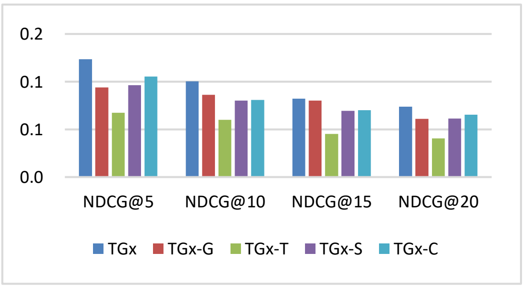
(c) Istanbul: Recall



(d) Istanbul: NDCG



(e) New York: Recall

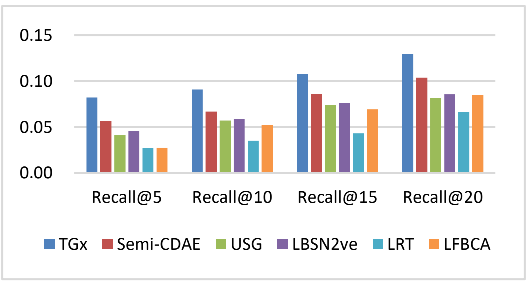


(f) New York: NDCG

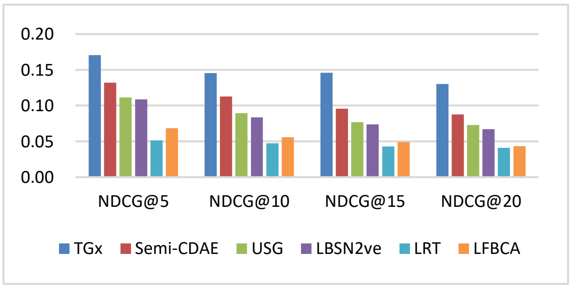
Fig.13 The accuracy of TGx and its simplified models for cold-start users.

Table 13 Cold-start experiments w.r.t. accuracy on New York Dataset

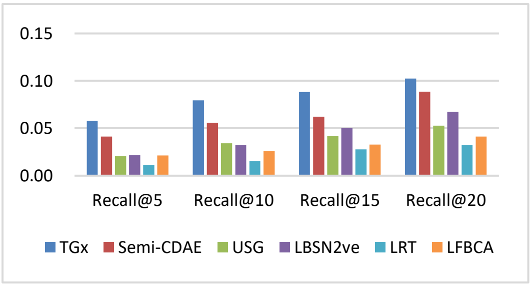
	Recall					NDCG				
Model	@5	@10	@15	@20	$\downarrow @k\%$	@5	@10	@15	@20	$\downarrow @k\%$
TGx	0.054	0.063	0.070	0.090	0.0	0.124	0.100	0.082	0.074	0.0
TGx-G	0.037	0.042	0.057	0.073	-25.5	0.094	0.086	0.080	0.061	-14.6
TGx-T	0.026	0.035	0.046	0.059	-41.1	0.068	0.060	0.045	0.040	-44.0
TGx-S	0.043	0.052	0.066	0.080	-13.5	0.096	0.080	0.069	0.061	-18.8
TGx-C	0.045	0.051	0.067	0.083	-11.7	0.105	0.081	0.070	0.065	-15.1
Model	@5	@10	@15	@20	$\uparrow @k\%$	@5	@10	@15	@20	$\uparrow @k\%$
TGx					231.5					235.9
Semi-CDAE	0.036	0.044	0.052	0.058	128.3	0.105	0.077	0.056	0.049	146.2
USG	0.021	0.029	0.036	0.046	56.9	0.044	0.034	0.032	0.029	24.9
LBSN2ve	0.026	0.029	0.040	0.052	73.4	0.090	0.066	0.058	0.053	135.1
LRT	0.018	0.019	0.020	0.026	0.0	0.042	0.030	0.025	0.019	0.0
LFBCA	0.018	0.026	0.031	0.038	34.8	0.047	0.041	0.036	0.028	34.2



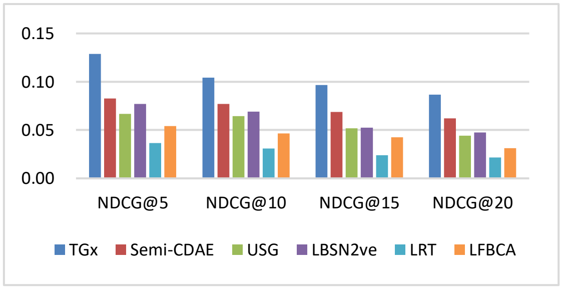
(a) Tokyo: Recall



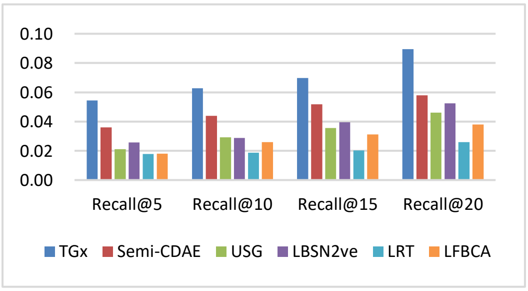
(b) Tokyo: NDCG



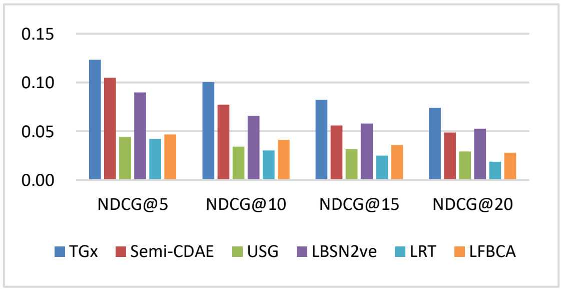
(c) Istanbul: Recall



(d) Istanbul: NDCG



(e) New York: Recall



(f) New York: NDCG

Fig.14 The accuracy of TGx and other advanced models for cold-start users.

E.5 Results of the Diversity Experiments

Table 14 Results of diversity experiments on Tokyo Dataset

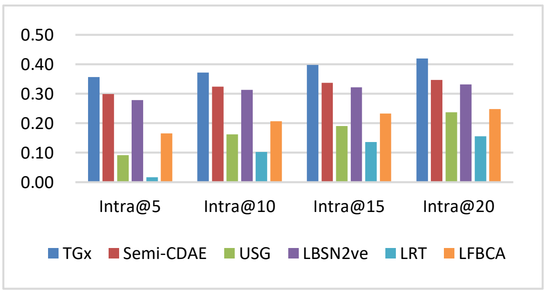
	Intra					Inter				
Model	@5	@10	@15	@20	$\overline{\uparrow @k}\%$	@5	@10	@15	@20	$\overline{\uparrow @k}\%$
warm-start										
TGx	0.357	0.372	0.399	0.419	208.69	0.996	0.997	0.998	0.998	79.00
Semi-CDAE	0.299	0.324	0.337	0.347	162.42	0.956	0.960	0.970	0.982	73.43
USG	0.092	0.162	0.191	0.237	50.44	0.784	0.799	0.859	0.891	48.93
LBSN2ve	0.278	0.313	0.322	0.331	151.84	0.942	0.954	0.966	0.977	72.07
LRT	0.017	0.102	0.136	0.155	0.00	0.493	0.544	0.592	0.615	0.00
LFBCA	0.165	0.207	0.233	0.248	77.64	0.909	0.937	0.942	0.968	68.18
out-of-town										
TGVx	0.359	0.368	0.400	0.417	193.70	0.991	0.996	0.997	0.997	78.67
Semi-CDAE	0.295	0.315	0.341	0.346	148.70	0.957	0.966	0.973	0.977	73.86
USG	0.086	0.156	0.186	0.233	40.57	0.779	0.790	0.866	0.880	48.28
LBSN2ve	0.275	0.305	0.326	0.333	139.49	0.948	0.953	0.962	0.972	72.09
LRT	0.018	0.111	0.145	0.152	0.00	0.498	0.546	0.586	0.610	0.00
LFBCA	0.168	0.206	0.234	0.253	70.90	0.909	0.932	0.941	0.974	68.32

Table 15 Results of diversity experiments on Istanbul Dataset

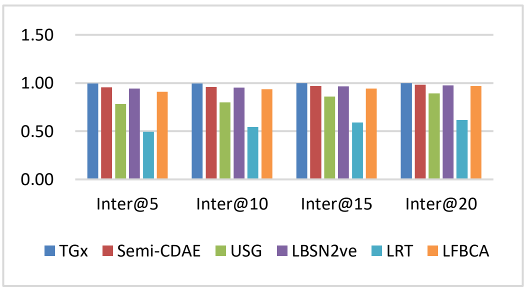
	Intra					Inter				
Model	@5	@10	@15	@20	$\overline{\uparrow @k}\%$	@5	@10	@15	@20	$\overline{\uparrow @k}\%$
warm-start										
TGx	0.399	0.431	0.442	0.471	91.84	0.996	0.997	0.997	0.998	48.39
Semi-CDAE	0.316	0.334	0.347	0.366	50.18	0.939	0.943	0.954	0.964	41.41
USG	0.291	0.347	0.379	0.385	53.52	0.873	0.888	0.898	0.909	32.71
LBSN2ve	0.278	0.331	0.337	0.357	43.09	0.916	0.925	0.947	0.958	39.34
LRT	0.196	0.212	0.244	0.261	0.00	0.631	0.663	0.686	0.712	0.00
LFBCA	0.307	0.328	0.345	0.382	49.65	0.957	0.961	0.966	0.975	43.55
out-of-town										
TGVx	0.402	0.425	0.443	0.463	92.64	0.994	0.997	0.998	0.998	49.60
Semi-CDAE	0.310	0.332	0.342	0.363	49.45	0.932	0.950	0.951	0.961	42.30
USG	0.293	0.348	0.376	0.389	55.18	0.881	0.894	0.898	0.913	34.54
LBSN2ve	0.276	0.337	0.345	0.345	44.36	0.917	0.927	0.951	0.960	40.83
LRT	0.189	0.212	0.243	0.265	0.00	0.616	0.659	0.689	0.708	0.00
LFBCA	0.302	0.331	0.340	0.382	50.12	0.958	0.955	0.973	0.974	44.82

Table 16 Results of diversity experiments on New York Dataset

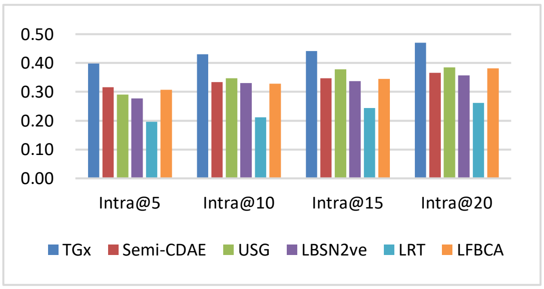
	Intra					Inter				
Model	@5	@10	@15	@20	$\overline{\uparrow @k}\%$	@5	@10	@15	@20	$\overline{\uparrow @k}\%$
warm-start										
TGx	0.455	0.508	0.526	0.536	96.56	0.985	0.987	0.989	0.992	51.21
Semi-CDAE	0.405	0.427	0.436	0.489	70.56	0.957	0.959	0.967	0.972	47.42
USG	0.325	0.359	0.388	0.418	44.24	0.814	0.847	0.851	0.871	29.27
LBSN2ve	0.439	0.485	0.518	0.529	91.09	0.948	0.950	0.957	0.961	45.91
LRT	0.222	0.257	0.268	0.287	0.00	0.608	0.642	0.663	0.708	0.00
LFBCA	0.399	0.423	0.438	0.472	68.12	0.945	0.951	0.960	0.965	46.11
out-of-town										
TGVx	0.468	0.501	0.530	0.537	99.75	0.981	0.985	0.992	0.993	51.70
Semi-CDAE	0.407	0.423	0.438	0.492	72.37	0.959	0.960	0.971	0.974	48.36
USG	0.321	0.366	0.388	0.425	46.34	0.811	0.838	0.846	0.868	29.02
LBSN2ve	0.441	0.493	0.512	0.522	92.86	0.946	0.954	0.958	0.959	46.61
LRT	0.218	0.252	0.269	0.285	0.00	0.610	0.635	0.661	0.705	0.00
LFBCA	0.397	0.420	0.431	0.465	67.97	0.942	0.958	0.960	0.964	46.86



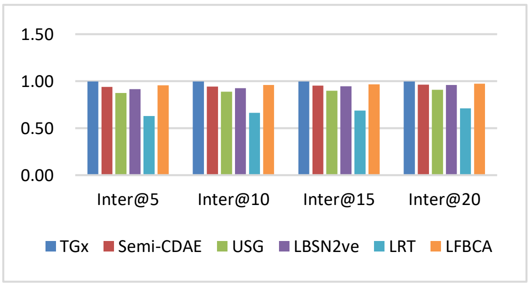
(a) Tokyo: Intra



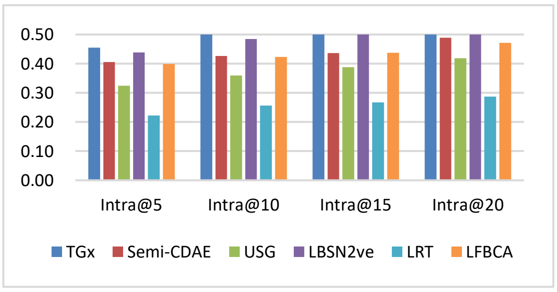
(b) Tokyo: Inter



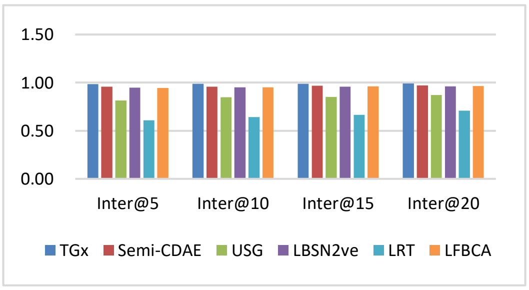
(c) Istanbul: Intra



(d) Istanbul: Inter

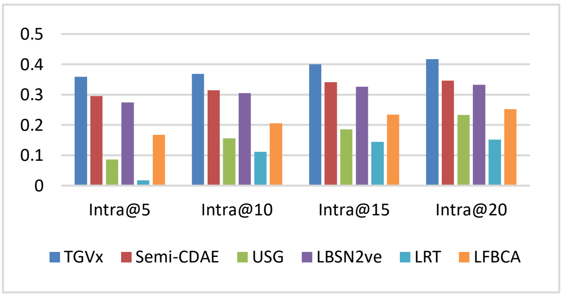


(e) New York: Intra

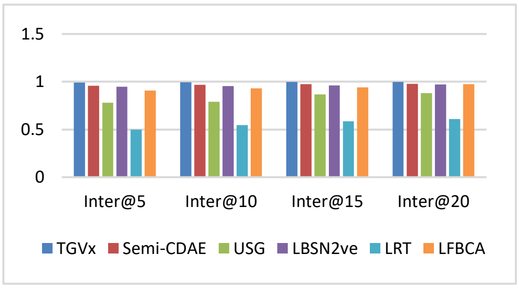


(f) New York: Inter

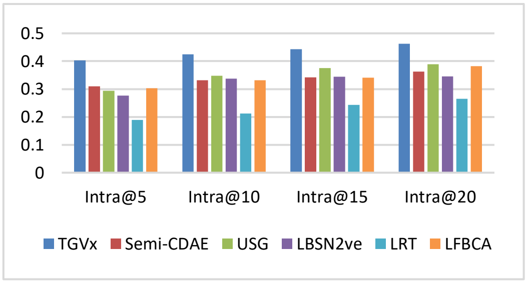
Fig.15 The diversity of TGx and other advanced models for warm-start users.



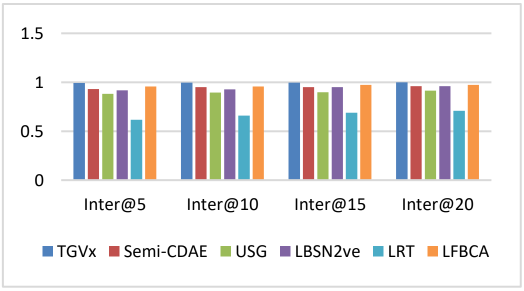
(a) Tokyo: Intra



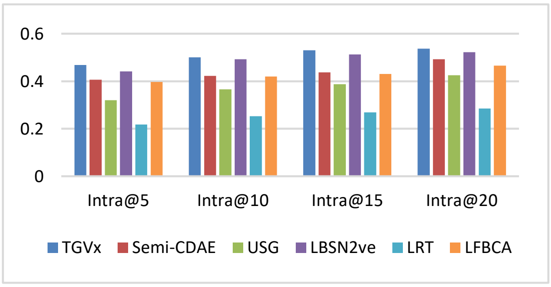
(b) Tokyo: Inter



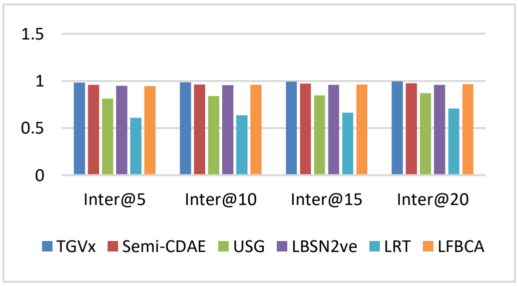
(c) Istanbul: Intra



(d) Istanbul: Inter



(e) New York: Intra



(f) New York: Inter

Fig.16 The diversity of TGVx and other advanced models for out-of-town users.

E.6 More Discussion

E.6.1 Recommendation Accuracy

The experimental results in Sections 5.3 and 5.4 show that, for **local and out-of-town** users, the six advanced models were sorted by accuracy as follows: TG(V)x> Semi-CDAE> LBSN2Vec> USG> LFBCA> LRT. The TG(V)x model had the highest accuracy, which was attributed to its outstanding multi-source information fusion ability and powerful high-order nonlinear user-POI interaction mining ability. The specific analysis is as follows.

In terms of **the fusion ability and utilization effect of multi-source heterogeneous information**: TG(V)x has the most advantage, followed by LBSN2Vec. For shallow models, the accuracy ranking results of LBSN2Vec>USG>LFBCA>LRT are largely due to their multi-source heterogeneous information fusion ability. LBSN2vec and TG(V)x respectively "indirectly" and "directly" fuse multi-source heterogeneous information into user-POI interactions. If collaborative and public information are not considered, which fusion method is more conducive to the improvement of recommendation performance, "indirect" or "direct"? To discuss this issue, we compare LBSN2vec with TGx's simplified version TGx-C, which considered the same information as LBSN2vec. Table 17 shows the improvement rate (%) of the accuracy of the TGx-C model compared to the LBSN2vec model, where $\overline{@k}$ denotes the average improvement rate. The comparison results prove that the “direct” integration of multi-source heterogeneous information into user-POI interactions is more conducive to improving the accuracy of POI recommendation.

Table 17 Compared with the LBSN2vec model, the improvement rate (%) of the accuracy of the TGx-C model

	Recall					**NDCG**				
	↑@5	↑@10	↑@15	↑@20	↑ $\overline{@k}$	↑@5	↑@10	↑@15	↑@20	↑ $\overline{@k}$
Tokyo										
warm-start	56.8	34.5	33.7	30.2	38.8	50.3	60.2	58.1	57.0	56.4
cold-start	57.0	30.6	29.5	45.8	40.7	44.7	58.9	59.8	75.7	59.8
Istanbul										
warm-start	39.9	47.6	29.8	26.2	35.9	23.3	25.3	39.9	18.9	26.8
cold-start	115.0	49.8	25.0	18.6	52.1	37.0	24.3	47.3	43.7	38.1
New York										
warm-start	53.4	27.6	18.2	30.2	32.3	30.7	40.3	27.0	31.8	32.5
cold-start	74.2	78.0	70.2	58.3	70.2	17.3	23.3	21.1	23.7	21.3

In the ability **to mine high-order nonlinear use-POI interactions**, TG(V)x has the most advantage, followed by Semi-CDAE, and then LBSN2Vec. TG(V)x and Semi-CDAE are POI deep recommendation models. LBSN2Vec, USG, LFBCA and LRT are shallow recommendation models. Semi-CDAE is inferior to LBSN2Vec in the fusion ability of multi-source heterogeneous information, and the former cannot integrate time information. However, the powerful high-order nonlinear mining capabilities of the DL network make up for this disadvantage to a certain extent, and Semi-CDAE is better than LBSN2Vec in terms of accuracy. Table 18 shows the improvement rate (%) of the accuracy of the TG(V)x model compared to Semi-CDAE. Overall, compared to warm-start users, the TG(V)x model improves the accuracy of POI recommendations for cold-start and out-of-town users to a greater extent.

Table 18 Compared with the Semi-CDAE model, the improvement rate (%) of the accuracy of the TG(V)x model

	Recall					**NDCG**				
	↑@5	↑@10	↑@15	↑@20	$\overline{\uparrow @k}$	↑@5	↑@10	↑@15	↑@20	$\overline{\uparrow @k}$
Tokyo										
warm-start	37.1	30.0	21.1	11.0	24.8	14.8	17.1	14.7	21.7	17.1
cold-start	45.1	35.9	25.3	25.1	32.9	29.4	29.1	52.3	48.2	39.8
out-of-town	44.0	52.4	55.5	47.8	49.9	53.0	56.7	54.4	70.2	58.6
Istanbul										
warm-start	38.2	44.8	44.4	20.1	36.9	26.9	17.0	21.9	20.9	21.7
cold-start	39.8	42.2	41.6	15.6	34.8	55.8	35.3	40.5	39.5	42.8
out-of-town	24.7	33.6	23.7	27.5	27.4	36.6	33.1	25.2	41.4	34.1
New York										
warm-start	40.6	40.0	28.8	30.7	35.0	25.9	37.1	48.9	46.7	39.7
cold-start	50.9	43.0	34.5	54.5	45.8	17.9	30.1	47.3	51.1	36.6
out-of-town	67.2	61.0	28.2	40.9	49.3	26.3	33.4	42.1	35.8	34.4

E.6.2 Recommendation Diversity

Compared to the LRT model, for warm-start users, on Inter@k, the Semi-CDAE model had the second highest rate of improvement, followed by the LBSN2Vec and LFBCA models, and the USG model had a lower level of improvement. On Intral@k, Semi-CDAE, LBSN2Vec, USG and LFBCA models were not completely consistent in the order of their improvement rates. For out-of-town users, on both Inter@k and Intral@k, Semi-CDAE, LBSN2Vec, USG and LFBCA models were not completely consistent in the order of their improvement rates. Table 19 shows the improvement rate (%) of the diversity of the TG(V)x model compared to the Semi-CDAE model. On the surface, the improvement rate of Inter@k of the TG(V)x model is not large, because Inter@5~Inter@20 of the Semi-CDAE model are all greater than 0.939. On this basis, the degree of improvement of TG(V)x's Inter@k is also obvious.

Table 19 Compared with the Semi-CDAE model, the improvement rate (%) of the diversity of the TG(V)x model

	Intra					Inter				
	↑@5	↑@10	↑@15	↑@20	$\overline{\uparrow @k}$	↑@5	↑@10	↑@15	↑@20	$\overline{\uparrow @k}$
Tokyo										
warm-start	19.49	14.96	18.16	20.83	18.36	4.23	3.81	2.86	1.65	3.14
out-of-town	21.47	16.82	17.32	20.48	19.02	3.48	3.01	2.38	2.03	2.73
Istanbul										
warm-start	26.02	28.98	27.41	28.67	27.77	6.01	5.64	4.49	3.49	4.91
out-of-town	30.02	28.26	29.55	27.62	28.86	6.61	4.88	4.96	3.92	5.09
New York										
warm-start	12.27	19.01	20.51	9.74	15.38	2.87	2.95	2.33	2.09	2.56
out-of-town	15.23	18.53	21.09	9.10	15.99	2.30	2.53	2.16	1.99	2.25

E.6.3 Nonlinear Design Vs. Linear Design

We conducted the "TG(V)x model vs. TF-IDF model" experiments aimed at answering "Does nonlinear design in TG(V)x have a clear performance advantage?". The linear TF-IDF model refers to the remaining part after removing the TG module from TG(V)x. According to Eq.(A18), the TF-IDF model can only calculate the preference value of POIs that user u has checked in, and cannot recommend POIs that he has not checked in. Therefore, there are great limitations in recommendation accuracy and diversity. To enable the TF-IDF model to recommend the unchecked POIs for user u, we add social influence on Eq.(A18), that is, the average preference value of all direct friends of the user to POI v. In addition, the TF-IDF model does not require training, and cannot use optimization functions to add social, collaborative, and public regularization terms.

Table 20 POI recommendation accuracy of TF-IDF model

	Recall				NDCG			
	@5	@10	@15	@20	@5	@10	@15	@20
Tokyo								
warm-start	0.0297	0.0301	0.0302	0.0302	0.0637	0.0635	0.0634	0.0625
out-of-town	0.0238	0.0239	0.0241	0.0242	0.0351	0.0349	0.0347	0.0348
Istanbul								
warm-start	0.0155	0.0155	0.0156	0.0156	0.0495	0.0494	0.0492	0.0488
out-of-town	0.0174	0.0176	0.0177	0.0177	0.0317	0.0317	0.0316	0.0313
New York								
warm-start	0.0174	0.0176	0.0177	0.0177	0.0317	0.0317	0.0316	0.0313
out-of-town	0.0116	0.0118	0.0119	0.0120	0.0138	0.0137	0.0136	0.0133

Table 20 gives the accuracy of the TF-IDF recommendation model. Whether it is a local or an out-of-town recommendation, the Recall@k and NDCG@k of the TF-IDF model do not change significantly with k, and we can barely observe the difference until the four decimal places are reserved. This is attributed to the insufficient mining ability of the linear TF-IDF model for the sparse check-in data. Comparing the experimental results of Table 20 and Table 5~

Table 7, it was not difficult to find that the nonlinear TG(V)x model was significantly higher than the linear TF-IDF model in terms of local and out-of-town recommendation accuracy. This also proved that the nonlinear design (i.e., TG module) does have a clear performance advantage.

E.6.4 Unequal Time Slots Vs. Equal Time Slots

According to Section 2.2 and Fig.1(a), we built a TGVx model based on 12 unequal time slots. Are the TG(V)s models based on unequal time slots better than those based on equal time slots? To answer this question, we conducted this experiment. 12 equal time slots starting at 00:00 with a granularity of 2 hours. Table 21 gave the POI recommendation accuracy based on the TG(V)s model for 12 equal time slots. Compared with the accuracy of the TG(V)s model with unequal time slots (see Table 5~

Table 7), the accuracy of local and out-of-town POI recommendations of the TG(V)s model with equal time slots decreased consistently. The effectiveness of unequal time slots was demonstrated, and the performance advantage it brings to the TG(V)x model.

Table 21 POI recommendation accuracy of TG(V)s model based on 12 equal time slots

	Recall				**NDCG**			
	@5	@10	@15	@20	@5	@10	@15	@20
Tokyo								
warm-start	0.095	0.110	0.131	0.162	0.202	0.193	0.176	0.162
out-of-town	0.054	0.065	0.084	0.111	0.070	0.068	0.059	0.049
Istanbul								
warm-start	0.076	0.090	0.113	0.141	0.172	0.159	0.143	0.130
out-of-town	0.043	0.053	0.064	0.088	0.069	0.057	0.049	0.041
New York								
warm-start	0.057	0.065	0.070	0.095	0.127	0.116	0.090	0.084
out-of-town	0.033	0.036	0.039	0.055	0.042	0.038	0.030	0.026

Appendix F Related Work

Our work is closely related to POI deep recommendations, heterogeneous information fusion in deep recommendations, and transfer learning for location-based services. We will review them next.

F.1 POI Deep Recommendations

Table 22 Common technology/model abbreviations in the paper

Abb.	Full name	Abb.	Full name
CD	Contrastive Divergence	CF	Collaborative Filtering
CNN	Convolutional Neural Network	DAE	Deep AutoEncoder
DL	Deep Learning	MF	Matrix Factorization
MN	Memory Network	MDBN	Multimodal Deep Belief Network
NDCG	Normalized Discounted Cumulative Gain	MLP	Multi-Layer Perceptron
PACE	Preference And Context Embedding	PMF	Probabilistic MF
RNN	Recurrent Neural Networks	SemiDAE	DAE based on SemiRBMs
SemiRBM	(Semi-Restricted) Boltzmann Machines	TGx	TGVx without V modular
Semi-CDAE	SemiDAE with the Conditional network	TEMN	Topic-Enhanced Memory Network
T-SemiDAE	SemiDAE with the Time-conditional network	WRMF	Weighted-Regularized MF
DMRL	Deep Multimodal Rank Learning	GNN	Graph Neural Network
PMGT	Pre-trained Multimodal Graph Transformer	GCN	Graph Convolution Network
MMGCN	Multimodal Graph Convolution Network	MGAT	Multimodal Graph Attention Network
GE	Graph Embedding	NCF	Neural Collaborative Filtering
MM-Rec	Multi-Modal news Recommendation	(D)TL	(Deep) Transfer Learning
EMCDR	Embedding and Mapping framework for Cross-Domain Recommendation		
LBSN2vec	a hyper-graph embedding approach designed specifically for LBSN data		
LFBCA	Location-Friendship Bookmark-Coloring Algorithm		
LRT	Location Recommendation framework with Temporal effects		
RecNet	a deep neural Network-based POI Recommendation framework		
SH-CDL	Spatial-Aware Hierarchical Collaborative Deep Learning model		
ST-TransRec	a Spatial and Textual Transfer learning, Recommendation		
SAE-NAD	a Self-Attentive Encoder and a Neighbor-Aware Decoder		
TGVx	Time and Geographic factors, out-o-town Visitors, and x time slot number		
USG	User preference, Social influence from friends and Geographical influence		
VCG	model incorporates Visual Contents and Geographical influence		
VGMF	Visual content and Geographical modeling enhanced MF model		
VPOI	Visual content enhanced POI recommendation model		

Deep Learning (DL) [45] has become one of the most important technologies in current recommender systems, and more and more advanced deep recommendation models are emerging from various commercial platforms [47,54,55]. According to the task type and the role of DL technology in recommendation models, we roughly divide the existing deep recommendation models into three categories: deep-sequential recommendation models [56], deep-embedding recommendation models [4], and deep-interaction recommendation models [7,46]. This section mainly takes the POI recommendation domain in LBSNs as the representative and introduces these three categories of deep recommendation models. This paper is dedicated to the research of a deep-interaction POI recommendation model. Table 22 lists common technology/model abbreviations in the paper.

F.1.1 Deep-sequential POI Recommendation

Sequential POI recommendations are dedicated to the prediction of sequential preference, and its goal is to mine the time sequence dependency of users checking in to POIs [17]. The research of DL technology on sequential POI recommendations is relatively new. This is mainly due to the powerful mining and representation capabilities of Recurrent Neural Networks (RNN) and its improved versions for sequence data [45], as well as the breakthrough and landing application of RNN technology in the field of natural language processing. After arranging user check-in records in chronological order, RNN technology can be used for sequential POI recommendations [57,58,59]. For example, Yang et al. [57] used RNN and gated recurrent units to model the order dependency of check-in POIs. Wang et al. [58] used attention based RNN to build a personalized route recommendation model., which not only mines the order dependency between POI check-ins, but also learns the time-varying vectorized representation of the user's mobile state. Zhao et al. [59] developed a spatiotemporal gated network, which uses an enhanced long- and short-term memory method to separately mine user short-term and long-term preferences to implement the next

POI recommendation. The deep-sequential POI recommendation is one of the most popular research directions at present. It has important theoretical research value, and good application and promotion prospects in tourism route recommendation, trajectory/path planning and other fields.

F.1.2 Deep-embedding POI Recommendation

Table 23 Comparison of DL-based conventional POI recommendation models

Year	Model	Scenario	Dynamic or Static	Factors	Learning Paradigm	Output Way	Fusion Method
Our	TGVx	L+O	Dynamic (T+S)	Tempral, Geo, Social, Semantic,Col, Public,	Un, Tran	one2all	DL (i.e, T-SemiDAE)
2018	Semi-CDAE [7]	L	Static	Geo, Social,Semantic	Un	one2all	DL (i.e., Semi-CDAE)
2017	VPOI [42]	L	Static	visual	Un	one2one	PMF
2017	PACE [32]	L	Static	Geo, Social	Semi	one2one	Multi-objective optimization
2019	TEMN [17]	L	Dynamic (T)	Tempral, Geo	Semi	one2one	Multi-objective optimization
2017	SH-CDL [4]	L+O	Dynamic (T+S)	Geo, Semantic, Pop	Supervised	one2one	MDBN
2020	ST-TransRec [5]	O	Dynamic (S)	Geo, Semantic	Semi, Tran	one2one	POI embedding
2018	SAE-NAD [44]	L	Static	Geo	Un	one2all	DL (i.e., DAE)
2018	RecNet [35]	L	Static	Geo, Semantic	Semi	one2one	POI embedding
2019	VCG [63]	L	Static	Geo, Social, Images	Un	one2one	WRMF
2020	VGMF [62]	L	Static	Geo, Images	Un	one2one	WRMF
2021	DMRL [65]	L	Dynamic (T)	Tempral, Geo, Semantic, visual	Supervised	one2one	DL(MLP)

In “Scenario” column, L denotes local and O denotes out-of-town.

In “Dynamic or Static” column, T denotes Temporal and S denotes Spacial.

In “Factors” column, Geo denotes Geographic, Pop denotes Popularity, Col denotes Collaborative.

In “Learning Paradigm” column, Un denotes Unsupervised, Semi denotes Semi-supervised, Tran denotes Transfer.

The conventional POI recommendation task for non-sequential preference prediction is generally called POI recommendation, and its goal is to mine complex user-POI interactions [17]. There are relatively few studies on the influence of DL technology on conventional POI recommendation. According to the role of DL technology in the model, the existing research can be roughly summarized into two categories: deep-embedding POI recommendations and deep-interaction POI recommendations. Table 23 and Table 24 compare DL-based conventional POI recommendation models and their DL techniques, respectively.

Table 24 Comparison of DL technologies in conventional POI recommendation models

Year	Model	DL's Type	DL's Task	DL's Output	DL's Learning Paradigm
our	TGVx	DAE	①,②	one2all	Un, Tran
2018	Semi-CDAE [7]	DAE	①,②	one2all	Un
2017	VPOI [42]	CNN	embedding Images	an image latent vector	Un, Tran
2017	PACE [32]	MLP	②	one2one	Supervised
2019	TEMN [17]	MN	embedding user-POI pair	a user-POI latent vector	Un
2017	SH-CDL [4]	DBN	①,embedding POIs	a POI latent vector	Un, Supervised
2020	ST-TransRec [5]	MLP	②	one2one	Supervised
2018	SAE-NAD [44]	DAE	①,②	one2all	Un
2018	Recent [35]	MLP	②	one2one	Supervised
2019	VCG [63]	CNN	embedding Images	an image latent vector	Un
2020	VGMF [62]	CNN	embedding Images	an image latent vector	Un
2021	DMRL [65]	CNN,LSTM,MLP	①,embedding Images and Text	an image latent vector a word latent vector	Supervised

In "DL's Task" column, ① denotes "fusing multimodal features", ② denotes "modeling user-POI interactions".

In "DL's Learning Paradigm" column, Un denotes Unsupervised, Tran denotes Transfer.

In **deep-embedding POI recommendations**, the role of DL technology is to develop embedding representations of users, POIs, images, etc. The output of the DL network is a latent vector representation, like the word2vec technology in the field of natural language processing to generate word vectors [60]. For example, the MDBN developed by Yin et al. [4] generate a unified semantic deep embedded representation for each POI through the effective integration of heterogeneous information of geography, semantics, and popularity. Wang et al. [42] proposed a Visual content enhanced POI recommendation model, i.e., VPOI, which for the first time integrates image information in the process of learning users and POI latent vectors and combines the PMF model to predict user preferences. The authors extracted the deep embedding representation of an image through the VGG16 [61] network in the CNN family and used the image information to alleviate the data sparsity issue. Inspired by the VPOI model, Liu et al. [62] proposed the Visual content and Geographical modeling enhanced MF model (VGMF). The authors considered the phenomenon of geographic aggregation, integrate image and geographic information in the process of learning users and POI latent vectors, and combined the WRMF model to predict user preferences, which further improves the accuracy of POI recommendations. Like VPOI and VGMF, the VCG model proposed by Zhang et al. [63] additionally considers social information and uses AlexNet [64] to extract the deep embedding representation of an image. The Deep Multimodal Rank Learning (DMRL) model proposed by Liao et al. [65] considers time, geography, images, and comment information, uses VGG16 to embed images, and trains LSTMAE model to embed comment text. Then, the image and text embedding vectors are fused by MLP as the latent vector of a POI.

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ denote the latent vectors of user u and POI v, respectively. The $\boldsymbol{u}$ and $\boldsymbol{v}$ obtained by a deep embedding POI recommendation model, and the preference value p_{uv} of u to v is obtained through the inner product $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}$. Essentially, it is a shallow linear model [44], not a deep mining of high-order nonlinear user-POI interactions.

Any existing deep embedding POI recommendation model outputs preferences in a "one2one" manner, and each execution can only obtain the preference of user u for one POI. Predicting u's preferences for all POIs in this way is often time-consuming. A commonly used compromise strategy is to select a POI candidate set for u, predict his/her preference for all POIs in the candidate set, and select top-k for recommendation after sorting [4,5]. Most of these models randomly select a small part of POIs to construct a candidate set, which is likely to reduce the diversity of recommendation results. A good recommender system should be able to recommend novel items (e.g., POIs) that users want but will not be easily found, and the system should have the diversity of recommendations. Diversity is the soul of personalized recommendation, and it is an effective guarantee for recommending different POIs for different users. Many scholars are committed to the research of recommendation diversity [66,67,68]. Diversity has many dimensions, of which there are two dimensions that have been widely studied: individual diversity and aggregate diversity. The **individual diversity** refers to the diversity of items in the recommendation list of individual users. The occasional recommendation brought about by individual diversity can help users develop new items, enhance user experience, and retain customers. The **aggregate diversity** refers to the diversity of the items in the recommended list collection of all users, which reflects the coverage of the item catalog. From a business perspective, aggregate diversity is important. For POI recommendations, low aggregate diversity means that some businesses will not be recommended. And every merchant has paid a fee to the platform, and it is obviously unreasonable to not get the opportunity to be recommended.

F.1.3 Deep-interaction POI Recommendation

Regarding the impact of DL technology on mining complex user-POI interactions, a systematic and comprehensive exploration has not yet been carried out. As far as we know, representative studies include Semi-CDAE [7], SAE-NAD [44], PACE [32], RecNet [35] and ST-TransRec [5]. All these studies tackle **deep-interaction POI recommendations**, in which the role of DL technology is to mine high-order nonlinear user-POI interactions and produce a user's preference for one or all POIs. According to the learning paradigm of DL network, the existing deep-interaction POI recommendation models are roughly divided into two categories: unsupervised and supervised learning. The former includes Semi-CDAE [7] and SAE-NAD [44] models, and the latter includes PACE [32], RecNet [35] and ST-TransRec [5] models. Although the PACE, RecNet and ST-TransRec models are semi-supervised learning, they belong to supervised learning when using DL network to learn user-POI interactions. Details are as follows:

Inspired by the Neural Collaborative Filtering (NCF) architecture proposed by He et al. [46], PACE, RecNet and ST-TransRec models use Multi-Layer Perceptron (MLP) to model users' POI preferences. The embedded user and POI input the MLP in pairs and output the POI preference in a "one2one" manner. The difference among them is the learning method of user and POI embedding vectors, and the purpose and mining method of multi-source information. PACE [32] uses word2vec [60] technology to learn the embedding vectors of users and POIs. By jointly optimizing the sum of the loss functions of the three tasks (i.e., user POI preference, user context, POI context), social and geographic information are integrated into the user's POI preference. RecNet [35] uses MF and vector concatenated methods to integrate social, geographic, and POI category information into the embedding vectors of users and POIs. ST-TransRec [5] first uses word2vec technology to learn the first-stage embedding vectors of users and POIs, and then optimizes these vectors based on transfer learning and joint cross-city geographic and POI category information. Unlike PACE and RecNet, which are only applicable to local recommendations, ST-TransRec is also applicable to out-of-town recommendations.

The supervised learning models PACE, RecNet and ST-TransRec regard POI preference prediction as a binary classification task. Suppose that the POIs signed in by the user are positive samples, which means they like them; POIs that have not been checked in are negative samples, which means they do not like them. Since the number of unchecked POIs is much greater than that of checked POIs, to prevent the issues of unbalanced categories, these three models only randomly select a part of negative examples. Most of the unchecked data will be "discarded", and the user-POI interaction information they implied will also be deleted from the analysis. Roughly assuming implicit feedback as explicit feedback ignores the nature of implicit feedback without negative feedback [36].

From the perspective of implicit feedback modeling, unsupervised learning models (Semi-CDAE and SAE-NAD) are more reasonable than supervised learning models (PACE, RecNet and ST-TransRec). This is because implicit feedback has no clear label information, e.g., likes/dislikes, or scores within a certain range of values. The value of implicit feedback (e.g., the number of user check-ins) indicates the degree of confidence, which is the credibility of the result of the check-in behavior. A larger feedback value does not indicate a higher degree of preference [36]. One of the advantages of unsupervised learning is that it does not require label information. The model/method under this paradigm mines the internal relationships or rules among samples, e.g., user-POI interactions. It is precisely because of the high degree of matching with implicit feedback that unsupervised learning has been favored by many POI recommendation researchers, and many classic models have been proposed, such as WRMF [33,41] and graph embedding POI [9,13,15] recommendation models. However, most of these unsupervised learning models are based on shallow linear methods to model the user's POI preferences and cannot dig deep into high-order nonlinear user-POI interactions.

Semi-CDAE and SAE-NAD are unsupervised deep-interaction POI recommendation models using Deep AutoEncoder (DAE). The units in the input (output) layer of DAE correspond to POIs one2one, and output POIs preferences in a "one2all" manner. That is, when a user u is recommended, DAE predict his/her preferences for all POIs once executed. The Semi-CDAE model [7] is based on stacked (Semi)RBM to implement DAE, where the internal connection of the SemiRBM visible layer represents the geographical similarity between POIs, and the SemiRBM hidden layer is connected to the social conditional network. Differently, the SAE-NAD model [44] is based on MLP to implement DAE, where an attention mechanism is added between the input layer and the first hidden layer of the encoder to improve personalization, and the POI neighbor perception decoder is used to simulate geographic influence. Semi-CDAE and SAE-NAD take advantage of the multi-level and structural diversity of a DL network to integrate geographic and/or social influences and map them to a more abstract high-level space to obtain more accurate POI recommendations.

Moreover, Semi-CDAE and SAE-NAD continue the preference-confidence division strategy of Hu et al. [36] to deal with implicit feedback (i.e., check-in data). Hu et al. believe that implicit feedback should be transformed into two paired quantities, namely preference and confidence. The division of preference-confidence has no parallelism in explicit feedback analysis, but it plays a key role in implicit feedback analysis. SAE-NAD adds a confidence matrix to the loss function, uses confidence as the weight of POI preference loss, and processes the confidence represented by individual user check-in frequency from the POI level. Differently, Semi-CDAE considered the confidence level represented by the check-in frequency from the POI category level (i.e., the authors called it the importance of the POI category to the user). The authors believe that the scale of check-in frequency is not comparable between different categories of POI. For example, in general, the user's check-in frequency for the life supermarket is often higher than the check-in frequency for the history museum, which does not mean that the user's preference for the life supermarket is higher than the preference for the history museum. They used TF-IDF technology to associate the user's check-in frequency for POIs with the POI category, and then obtained the category-level POI preference and used it as the input of the Semi-CDAE model.

F.1.4 TGVx Model Vs. Existing DL-Based POI Recommended Models

This paper studies conventional POI recommendation task and proposes a dynamic deep-interaction POI recommendation model—TGVx, which has obvious differences and improvements from the existing DL-based POI recommendation models. For a detailed comparison, see Table 23 and Table 24. Here, we only introduce a few main aspects:

POI recommendation scenarios and dynamic perception capabilities. As far as we know, only the TGVx and SH-CDL [4] models consider the spatiotemporal dynamics of user check-in behavior and are suitable for providing POI recommendations for local and out-of-town users. SH-CDL uses public preferences with time characteristics to obtain high-quality POI latent vectors. Differently, TGVx “directly” affects the time factor in the user-POI interactions. On the one hand, it matches the TGV model in the corresponding time slot according to the current (or user pre-check-in) time. On the other hand, each TGV model receives time influence through T-SemiDAE's conditional network. ST-TransRec [5] considers the spatial dynamics of the user's check-in behavior. Although it is also suitable for local and out-of-town recommendations, it ignores the temporal dynamics. On the contrary, Topic-Enhanced Memory Network (TEMN) [17] considers the temporal dynamics and ignores the spatial dynamics. It is suitable for local and sequence recommendation tasks. In the TEMN model, Zhou et al. used the temporal Latent Dirichlet Allocation method to capture the global POI preference, and the time influence is embedded in the user latent vector. They also use the Memory Network (MN) to capture local POI preferences and learn the relational embedding vector of each pair of <user, POI>. Although this vector represents the complex interaction between a user and a POI, it cannot “directly” obtain the user's POI preferences. The remaining DL-based POI recommended models are pure static models.

Output mode of POI recommendation models. TGVx, Semi-CDAE [7] and SAE-NAD [44] adopt a "one2all" output mode: when recommending to user u, the model can predict his/her preferences for all POIs once the model is executed. This mode is attributed to the structural characteristics of the multiple units of the DAE output layer. Considering the potential of improving recommendation diversity, we adopt the "one2all" output mode when designing TGVx. We also borrowed ideas from Semi-CDAE and SAE-NAD to mine user preferences under the unsupervised learning paradigm. Different from our TGVx model, Semi-CDAE and SAE-NAD are purely static POI recommendation models, which cannot solve the spatiotemporal drift of user interests. The other DL-based POI recommendation models adopt a "one2one" output mode.

The structure of the POI recommendation models. Both DMRL [65] and TGVx have parallel structures based on different time slots. Differently, DMRL belongs to the deep-embedding recommendations, while TGVx is a deep-interaction recommendation model. Although the idea of parallel structure based on different time slots is very simple, it is difficult to implement. The main reason is that the user-POI matrix will become more sparser after dividing the check-in data set by time slot [6]. To solve this issue, the conditional network of T-SemiDAE in TGVx receives users' POI preferences over similar time slots.

DL technology used in POI recommendation models. Although TGVx is inspired by Semi-CDAE [7], TGVx has been improved in DL technology. Users in LBSNs must personally check in to POIs to experience them. For users to conduct such physical world activities, they need to consider factors such as geographic distance and allowable time [2,3,17]. In terms of the impact on the quality of POI recommendations, the special factors of LBSNs (i.e., geographic and temporal information) are more important than the common factors of social networks (e.g., semantic, social, collaborative information, etc.) [6,14]. Therefore, when designing the DL network, we let the main body of T-SemiDAE (SemiDAE and conditional network) carry and integrate geographic and time information and incorporate auxiliary information such as collaborative users and social interactions into the training process of T-SemiDAE. Specifically, T-SemiDAE uses geographic similarity as the weight of the intra-layer connections of the SemiRBM visible layer, which is the same as the Semi-CDAE model. The conditional network receives the POI preferences of users in similar time slots and connects to the visible and hidden layers of SemiRBM, which is different from the Semi-CDAE model. We propose an optimization algorithm based on collaborative-social regularization term to learn T-SemiDAE. Differently, Semi-CDAE's conditional network receives social information and connects to the visible layer of SemiRBM. We also improved the geographic similarity model of Semi-CDAE and proposed a probability model of geographic similarity to improve the robustness of the recommendation model. Moreover, SAE-NAD [44] uses MLP-based DAE, and its encoder and decoder are not mirror-symmetrical. An attention mechanism is added between the input layer and the first hidden layer to improve personalized recommendation capabilities, and the output layer adds geographic distance information. Differently, T-SemiDAE and Semi-CDAE both use SemiDAE, the encoder and decoder are mirrored and symmetrical, and the input layer and output layer both receive geographic influence. Compared with Semi-CDAE and SAE-NAD, T-SemiDAE has better scalability in integrating more information. The network can be expanded horizontally in a conditional layer or added to the optimization algorithm in the form of a regularization term.

F.2 Heterogeneous Information Fusion in Deep Recommendation

DL promotes the development of hybrid recommendation methods, which can combine multi-source heterogeneous (i.e., multi-modal) information and collaborative filtering effects for better recommendation. Multimodal deep recommendation has become one of the most important research areas in multimodal applications. Early multimodal deep recommendation methods mainly use the **early fusion mechanism**, which uniformly add arbitrary continuous and discrete original features to the input layer of a single DL network and use a multi-layer network to automatically learn multimodal information [4]. Due to the difference and complexity of the numerical types and distributions of multi-source heterogeneous features, it is extremely difficult for the early fusion mechanism to meet the increasingly complex recommendation application requirements.

To overcome the shortcomings of the early fusion mechanism, researchers have developed the **late fusion mechanism**, which aims to learn the latent representations for each specific modality independently, and then "connects" these latent representations to form multimodal inputs [4]. The late fusion mechanism can individually select a latent variable model that is friendly to a particular modality. In addition, "connection" can be simple vector concatenation [69], addition [70,71], etc., or shared space and/or neural network methods [4,65]. For example, Shen et al. [72] studied the application of multimodal social media content for music recommendation. The authors first obtain hand-crafted and deep features for each modality (i.e., text and image), then propose an Attentive Multimodal Autoencoder (AMAE) for cross-modal latent representation learning, and finally concatenate the embedding vectors of all modalities to get latent vectors of users. Wu et al. [69] proposed the Multi-Modal news Recommendation (MM-Rec) model, which uses the ViLBERT network [73] to learn the embedded representations of news headlines and images with intrinsic correlation, and then concatenates the embedding vectors of text and images to obtain latent vectors of items. The authors also proposed a cross-modal candidate-aware attention network to obtain latent vectors of users. Wei et al. [71] proposed a Multimodal Graph Convolution Network (MMGCN) recommendation framework to build a user-item bipartite graph in each modality (i.e., image, sound, and style). The topological structure and message passing mechanism of a Graph Convolution Network (GCN) are used to enrich the embedding representations of user/item nodes in each modality, and then these embedding representations in all modalities were fused by addition operation. Tao et al. [74] proposed a Multimodal Graph Attention Network (MGAT), which improves the independent and modal-specific graphs in MMGCN into multimodal interaction graphs. MGAT also utilizes a gated attention mechanism to identify the different degrees of influence of different modalities on user preferences. Liu et al. [75] proposed a Pre-trained Multimodal Graph Transformer (PMGT) model. The node embedding representations of the item-item bipartite graph are initialized using multimodal (i.e., image, text, user purchase history) features, then the multimodal representations of nodes are enhanced by a Graph Neural Network (GNN), and finally the downstream recommendation task is completed by the NCF [46] model. MMGCN, MGAT, PMGT are all feature-based multimodal graph representation methods [76], which add multimodal features to node features. Differently, Sun et al. [77] proposed a node-based multimodal graph representation method [78] for recommender systems, called Multi-modal Knowledge Graph Attention Network (MKGAT). The knowledge graph constructs by MKGAT included not only traditional user and item nodes, but also modal-specific nodes such as images and texts, and then the node representation is enhanced by GNN with attention mechanism.

In addition, the multimodal fusion techniques in MF recommendations are still applicable to deep recommendation models. For example, VPOI [42], PACE [32], and TEMN [17] models use multi-objective optimization and regularization methods to fuse multi-modal information. Song et al. [79] proposed a Neuro-stylist model based on a dual fusion mechanism. On the one hand, multiple cross-modal autoencoders are utilized to encode image and text information, and on the other hand, the recommender and all modality-specific autoencoders are jointly trained by a multi-objective optimization method.

Different from the above-mentioned multimodal deep recommendation model, the heterogeneous information fusion technology used by TGVx's T-SemiDAE network belongs to a hybrid method. The input layer of T-SemiDAE receives POI-category information, the conditional network receives time information, the input/output layer inner weight connections of T-SemiDAE carry geographic information, and the regular terms in the optimization method coordinate social, collaborative, and public information. At present, the TGVx model does not integrate comment-based textual information and image-based visual information, which will be one of our future research works.

F.3 Transfer Learning for Location-Based Service

With the development of machine learning and the needs of practical applications, it is expected that machines can intelligently use previously learned knowledge to perform new tasks and/or solve new problems faster and better. Transfer Learning (TL) came into play. Transfer learning aims to extract knowledge from one or more source domains and transfer that knowledge to the target domain [18,80]. Transfer learning focuses on solving tasks/problems in the target domain, and the roles of source and target domains are asymmetric [18].

To solve the problem of insufficient data volume and/or computing power, more and more researchers are devoted to the research of Deep Transfer Learning (DTL) [81]. We judge the role of DL in DTL from two levels: "what to transfer" (at the data and feature level) and "how to transfer" (at the technical level). Then, DTL is roughly divided into two types. One type is the network/model-based DTL method that belongs to the "what to transfer" level, in which DL plays the role of knowledge (i.e., "what to transfer"), and the pre-trained DL network from the source domain is used as the transferred object [81]. Many of the deep recommendation models introduced above involve network-based DTL. For example, VPOI [42] and DMRL [65] use the pre-trained model VGG16, and MM-Rec [69] uses the pre-trained model ViLBERT. Another type is the mapping based DTL method that belongs to the "how to transfer" level. DL as a mapping function realizes the transfer of knowledge from the source domain to the target domain. The representative research is the Embedding and Mapping framework for Cross-Domain Recommendation (EMCDR) proposed by Man et al. [82]. EMCDR uses a DL network (e.g., MLP) to map latent vectors from the source domain into the target domain. The key to the success of EMCDR lies in the good generalization of the DL mapping network. To this end, Zhu et al. [83] introduced meta-learning technology, proposed a Transfer-Meta framework for Cross-Domain Recommendation (TMCDR), and then implemented the personalized meta-learning function [84].

Recently, (Deep) Transfer Learning ((D)TL) based on location services has received a lot of attention. In the field of smart cities and urban computing, Wang et al. [85] explored the problem of "how to develop a new smart city service with limited data?", and proposed the paradigm of urban transfer learning, the idea of which is to accelerate the development of smart cities through transfer learning. Wei et al. [86] proposed the FLexible multimOdal tRAnsfer Learning (FLORAL) method for the computational task of cold-start cities, which are cities where services and infrastructure have just been built or are not yet ready. FLORAL transfers knowledge from source cities with abundant multimodal and labeled data to target cities of the same type but with scarce labeled and multimodal data. Chen et al. [87] explored the city-transfer problem for the POI search recommendation task of Baidu Maps, and propose a Curriculum Hardness Aware Meta-Learning (CHAML) framework to transfer knowledge from multiple source cities with abundant check-in records to multiple cold-start cities with scarce check-in records.

In the field of POI recommendation, for the local recommendation scenario, Farseev et al. [88,89] proposed several cross-platform POI category recommendation methods to solve the problem of data sparsity. However, these methods require every user to have posted comments and images on all involved platforms (e.g., Twitter, Foursquare, and Instagram considered in [88,89]), and the number of users who meet this condition and the amount of contents they generate are usually insufficient to support personalized POI recommendations. Wang et al. [90] explored how to recommend items from information domain platforms (e.g., Trip) for users of social domain platforms (e.g., Facebook and Twitter), and propose a Neural Social Collaborative Ranking (NSCR) method to connect the information domain and social domain by bridging users (i.e., users that overlap across domains). NSCR first uses (Graph Embedding, GE) to represent user and item nodes in the user-POI graph of the information domain, and fixes the latent vector of bridging users, and then uses GE to represent the non-bridging user nodes in the user-user graph of the social domain to realize cross-domain knowledge transfer. Considering the cross-platform user privacy protection issue, Gao et al. [91] proposed a Confidence aware Collective Matrix Factorization (CCMF) method to connect the source and target platforms through bridging items (i.e., POIs overlapping across platforms). The user-POI interaction data of the source platform (e.g., Foursquare) is encrypted and then migrated to the target platform (e.g., Wechat) for POI recommendations.

For the out-of-town recommendation scenario, Li et al. [19] considered the differences and similarities of different cities and proposed a Common-topic Transfer Learning Model (CTLM), which separates city-specific topics that reflect city differences from shared topics that reflect city similarities. When CTLM makes out-of-town recommendations, only shared topics and target city-specific topics are used, which avoids the mismatch problem that is easy to occur in cross-city knowledge transfer. Similarly, Ding et al. [20] proposed a User Interest

Drift and Transfer (UIDT) model for out-of-town POI recommendation, which uses codebook [92] technology to transfer city-independent user interests. Xin et al. [93] believed that the check-in behavior of out-of-town users not only depends on the user's hometown preference but also on the user's travel intention and then propose the TRAvel-INtention-aware Out-of-town Recommendation (TRAINOR) model. The model uses a mapping based DTL method to map the user's hometown preference vector into the MLP network and outputs the transferred vector.

Both our TGVx and these TL-based out-of-town recommendation models use users' inherent check-in preferences as bridging features (i.e., cross-domain "overlapping" features) to connect source and target cities. The difference lies in the method of "how to transfer", and the extraction method of user's inherent check-in preference related to the basic recommender. CTLM is a topic-based model that reflects users' inherent preferences through topics shared across cities. UIDT is a Matrix Factorization-based model, where users share their inherent preference features in latent vectors of different cities. TRAINOR hides the user's inherent preference in the user's hometown preference vector and transfers it through MLP mapping. Differently, TGVx hides the user's inherent preference in the user's check-in records in his hometown and uses GE to find the most similar POI across cities and form a pseudo check-in record to realize the transfer.

Appendix G References

[1] Yang D, Qu B, Cudre-Mauroux P (2021) Location-centric social media analytics: challenges and opportunities for smart cities. *IEEE Intell. Syst.* 36(5):3–10.

[2] Bao J, Zheng Y, Mokbel M F (2012) Location-based and preference-aware recommendation

using sparse geo-social networking data. *Proc. 20th Int. Conf. on Advances in Geographic Information Systems*, Redondo Beach California, 2012, 199–208.

[3] Yin H, Zhou X, Cui B, Wang H, Zheng K, Nguyen QVH (2016) Adapting to user interest drift for POI recommendation. *IEEE T. Knowl. Data En.* 28(10):2566–2581.

[4] Yin H, Wang W, Wang H, Chen L, Zhou X (2017) Spatial-aware hierarchical collaborative deep learning for POI recommendation. *IEEE T. Knowl. Data En.* 29(11):2537–2551.

[5] Li D, Gong Z (2022) A deep neural network for crossing-city POI recommendations. *IEEE T. Knowl. Data En.* 34(8):3536–3548.

[6] Yuan Q, C Gao, Ma Z, Sun A, Magnenat-Thalmann N (2013) Time-aware point-of-interest recommendation. *Proc. 36th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Dublin Ireland, 2013, 363–372.

[7] Guo J, Zhang W, Fan W, Li W (2018) Combining geographical and social influences with deep learning for personalized point-of-interest recommendation. *J. Manage. Inform. Syst.* 35(4):1121–1153.

[8] Tobler WR (1970) A computer movie simulating urban growth in the Detroit region. *Econ. Geogr.* 46:234–240.

[9] Xie M, Yin H, Wang H, Xu F, Chen W, Wang S (2016) Learning graph-based POI embedding for location-based recommendation. *Proc. 25th ACM Int. on Conf. on Information and Knowledge Management*, Indianapolis Indiana USA, 2016, 15–24.

[10] Gao H, Tang J, Hu X, Liu H (2013) Exploring temporal effects for location recommendation on location-based social networks. *Proc. 7th ACM Conf. on Recommender Systems*, Hong Kong China, 2013, 93–100.

[11] Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. *Proc. 17th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, San Diego California USA, 2011, 1082–1090.

[12] Quan Y, Gao C, Sun A (2014) Graph-based point-of-interest recommendation with geographical and temporal influences. *Proc. 23rd ACM Int. Conf. on Information and Knowledge Management*, Shanghai China, 2014, 659–668.

[13] Yang D, Qu B, Yang J, Cudre-Mauroux P (2019) Revisiting user mobility and social relationships in LBSNs: a hypergraph embedding approach. *Proc. 28th Int. Conf. on World Wide Web*, San Francisco CA USA, 2019, 2147–2157.

[14] Ye M, Yin P, Lee WC, Lee DL (2011) Exploiting geographical influence for collaborative point-of-interest recommendation. *Proc. 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Beijing China, 2011, 325–334.

[15] Wang H, Terrovitis M, Mamoulis N (2013) Location recommendation in location-based social networks using user check-in data. *Proc. 21st ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, Orlando Florida, 2013, 374–383.

[16] Li R, Wang S, Deng H, Wang R, Chang KCC (2012) Towards social user profiling: unified and discriminative influence model for inferring home locations. *Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Beijing China, 2012, 1023–1031.

[17] Zhou X, Mascolo C, Zhao Z (2019) Topic-enhanced memory networks for personalised point-of-interest recommendation. *Proc. 25th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*, 2019, Anchorage AK USA, 3018–3028.

[18] Pan SJ, Yang Q (2010) A Survey on transfer learning. *IEEE T. Knowl. Data En.* 22(10): 1345–

1359.

[19] Li D, Gong Z, Zhang D (2019) A common topic transfer learning model for crossing city POI recommendations. *IEEE T. Cybernetics* 49(12): 4282–4295.

[20] Ding J, Yu G, Li Y, Jin D, Gao H (2019) Learning from hometown and current city: cross-city POI recommendation via interest drift and transfer learning. *IMWUT* 3(4):1–28.

[21] Grover A, Leskovec J. (2016) Node2vec: scalable feature learning for networks. *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, San Francisco California USA, 2016, 855–864.

[22] Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. *Proc. 27th Int. Conf. on Machine Learning*, Haifa Israel, 2010, 807–814.

[23] Welling M, Hinton GE (2002) A new learning algorithm for mean field Boltzmann machines. *Proc. Int. Conf. on Artificial Neural Networks*, Madrid Spain, 2002, 351–357.

[24] Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. *Proc. 14th Int. Conf. on World Wide Web*, Chiba Japan, 2005, 22–32.

[25] Zhou T, Su RQ, Liu RR, Jiang LL, Wang BH, Zhang YC (2009) Ultra accurate personalized recommendation via eliminating redundant correlations. *New J. Phys.* 11(12):123008.

[26] Guo X, Chen G, Wang C, Wei Q, Zhang Z (2021) Calibration of voting-based helpfulness measurement for online reviews: an iterative Bayesian probability approach. *INFORMS J. Comput.* 33(1):246–261.

[27] Zhang J, Wang C, Chen G (2021) A review selection method for finding an informative subset from online reviews. *INFORMS J. Comput.* 33(1):280–299.

[28] Aliannejadi M, Rafailidis D, Crestani F (2020) A joint two-phase time-sensitive regularized collaborative ranking model for point of interest recommendation. *IEEE T. Knowl. Data En.* 32(6):1050–1063.

[29] Qian T, Liu B, Nguyen QVH, Yin H (2019) Spatiotemporal representation learning for translation-based POI recommendation. *ACM T. Inform. Syst.* 7(2):1–24.

[30] Bell RM, Koren Y (2007) Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter* 9(2):75–79.

[31] Li X, Cong G, Li XL, Pham TAN, Krishnaswamy S (2015). Rank-GeoFM: a ranking based geographical factorization method for point of interest recommendation. *Proc. 38th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Santiago Chile, 2015, 433–442.

[32] Yang C, Bai L, Zhang C, Yuan Q, Han J. Bridging collaborative filtering and semi-supervised learning: a neural approach for POI recommendation. *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Halifax NS Canada, 2017, 1245–1254.

[33] Lian D, Zhao C, Xie X, Sun G, Chen E, Rui Y (2014) GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, New York New York USA, 2014, 831–840.

[34] Li H, Ge Y, Hong R, Zhu H (2016) Point-of-Interest recommendations: learning potential check-ins from friends. *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, San Francisco California USA, 2016, 975–984.

[35] Ding R, Chen Z (2018) RecNet: A deep neural network for personalized POI recommendation in location-based social networks. *Int. J. Geogr. Inf. Sci.* 32(8):1631–1648.

[36] Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. *Proc. 8th IEEE Int. Conf. on Data Mining*, Pisa Italy, 2008, 263–272.

[37] Ference G, Ye M, Lee WC (2013) Location recommendation for out-of-town users in location-based social networks. *Proc. 22nd ACM Int. Conf. on Information and Knowledge Management*, San Francisco California USA, 2013, 721–726.

[38] Wang W, Yin H, Chen L, Sun Y, Sadiq S, Zhou X (2015) Geo-SAGE: A geographical sparse additive generative model for spatial item recommendation. *Proc. 21st ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Sydney NSW Australia, 2015, 1255–1264.

[39] Wang W, Yin H, Chen L, Sun Y, Sadiq S, Zhou X (2017) ST-SAGE: a spatial-temporal sparse additive generative model for spatial item recommendation. *ACM T. Intel. Syst. Tec.* 8(3):1–25.

[40] Koren Y, Research Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

[41] Lian D, Zheng K, Ge Y, Cao L, Chen E, Xie X (2018) GeoMF++: scalable location recommendation via joint geographical modeling and matrix factorization. *ACM T. Inform. Syst.* 36(3):1–29.

[42] Wang S, Wang Y, Tang J, Shu K, Ranganath S, Liu H (2017) What your images reveal: exploiting visual contents for point-of-interest recommendation. *Proc. 26th Int. Conf. on World Wide Web*, Perth Australia, 2017, 391–400.

[43] Davtalab M, Alesheikh AA (2021) A POI recommendation approach integrating social spatio-temporal information into probabilistic matrix factorization. *Knowl. Inf. Syst.* 63:65–85.

[44] Ma C, Zhang Y, Wang Q, Liu X (2018) Point-of-interest recommendation: exploiting self-attentive autoencoders with neighbor-aware influence. *Proc 27th ACM Int. Conf. on Information and Knowledge Management*, Torino Italy, 2018, 697–706.

[45] LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521:436–444.

[46] He X, Liao L, Zhang H, Nie L, Hu X, Chua TS. Neural collaborative filtering. *Proc. 26th Int. Conf. on World Wide Web*, Perth Australia, 2017, 173–182.

[47] Zeynep B, Yurekli A, Bilge A, Kaleli C (2019) A review on deep learning for recommender systems: challenges and remedies. *Artif. Intell. Rev.* 52(1):1–37.

[48] Hinton GE (2012) A practical guide to training restricted Boltzmann machines. Montavon G., Orr G.B., Müller KR, eds. *Neural Networks: Tricks of the Trade* (Springer, Berlin, Heidelberg), 599–619.

[49] Dahl GE, Sainath TN, Hinton GE (2013) Improving deep neural networks for LVCSR using rectified linear units and dropout. *Proc. 2013 IEEE Int. Conf. on Acoustics, Speech and Checkal Processing*, Vancouver BC Canada, 2013, 8609–8613.

[50] Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) LINE: large-scale information network embedding. *Proc. 24th Int. Conf. on World Wide Web*, Florence Italy, 2015, 1067–1077.

[51] Dong Y, Chawla NV, Swami A (2017) Metapath2vec: scalable representation learning for heterogeneous networks. *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Halifax NS Canada, 2017, 135–144.

[52] Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. *Proc. 24th Int. Conf. on Machine learning*, 2007,791–798.

[53] Hinton GE, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. *Science* 313(5786): 504–507.

[54] Covington P, Adams J, Sargin E (2016) Deep neural networks for YouTube recommendations. *Proc. 10th ACM Conf. on Recommender Systems, Boston MA USA*, 2016, 191–198.

[55] Wu L, He X, Wang X, Zhang K, Wang M (2022) A survey on accuracy-oriented neural recommendation: from collaborative filtering to information-rich recommendation. *IEEE T. Knowl. Data En.* (Early Access).

[56] Yang D, Fankhauser B, Rosso P, Cudre-Mauroux P (2020) Location prediction over sparse user mobility traces using RNNs: Flashback in hidden states! *Pro. 29th Int. Joint Conf. on Artificial Intelligence*, Yokohama Yokohama Japan, 2020, 2184–2190.

[57] Yang C, Sun M, Zhao WX, Liu Z, Chang EY (2017) A neural network approach to jointly modeling social networks and mobile trajectories. *ACM T. Inform. Syst.* 35(4):1–28.

[58] Wang J, Wu N, Zhao WX, Peng F, Lin X (2019) Empowering a search algorithms with neural networks for personalized route recommendation. *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Anchorage AK USA, 2019, 539–547.

[59] Zhao P, Zhu H, Liu Y, Xu J, Li Z, Zhuang F, Sheng VS, Zhou X (2019) Where to go next: a spatio-temporal gated network for next poi recommendation. *Proc. 33rd AAAI Conf. on Artificial Intelligence*, Honolulu Hi, 2019, 5877–5884.

[60] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. *Proc. 26th Int. Conf. on Neural Information Processing Systems*, Lake Tahoe Nevada, 2013, 3111–3119.

[61] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. *Proc. 3rd Int. Conf. on Learning Representations*, San Diego CA USA, 2015, 1–14.

[62] Liu B, Meng Q, Zhang H, Xu K, Cao J (2020) VGMF: visual contents and geographical influence enhanced point-of-interest recommendation in location-based social network. *T. Emerg. Telecommun. T.*, Special Issue Article, 1–17.

[63] Zhang Z, Zou C, Ding R, Chen Z (2019) VCG: exploiting visual contents and geographical influence for point-of-interest recommendation. *Neurocomputing* 357(10): 53–65.

[64] Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60(6): 84–90.

[65] Liao J, Liu T, Yin H, Chen T, Wang J, Wang Y (2021) An integrated model based on deep multimodal and rank learning for point-of-interest recommendation. *World Wide Web* 24:631–655.

[66] Adomavicius G, Kwon Y (2014) Optimization-based approaches for maximizing aggregate recommendation diversity. *INFORMS J. Comput.* 26(2):351–369.

[67] Muter I, Aytekin T (2017) Incorporating aggregate diversity in recommender systems using scalable optimization approaches. *INFORMS J. Comput.* 29(3): 405–421.

[68] Çanakoğlu E, Muter İ, Aytekin T (2021) Integrating individual and aggregate diversity in top-n recommendation. *INFORMS J. Comput.* 33(1): 300–318.

[69] Wu C, Wu F, Qi T, Huang Y (2022) MM-Rec: multimodal news recommendation. *Proc 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid Spain, 2022, 2560–2564.

[70] Zhang F, Yuan NJ, Lian D, Xie X, Ma WY (2016) Collaborative knowledge base embedding for recommender systems. *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, San Francisco CA USA, 2016, 353–362.

[71] Wei Y, Wang X, Nie L, He X, Hong R, Chua TS (2019) MMGCN: multi-modal graph convolution network for personalized recommendation of micro-video. *Proc. 27th ACM Int. Conf. on Multimedia*, Nice France, 2019, 1437–1445.

[72] Shen T, Jiay J, Li Y, Wang H, Chen B (2020) Enhancing music recommendation with social media content: an attentive multimodal autoencoder approach. *Proc. Int. Joint Conf. on Neural Networks*, Glasgow UK, 2020, 1–8.

[73] Lu J, Batra D, Parikh D, Lee S (2019) Vilbert: pretraining task-agnostic visio-linguistic representations for vision-and-language tasks. *Proc. 33rd Conf. on Neural Information Processing Systems*, Vancouver Canada, 2019, 13–23.

[74] Tao Z, Wei Y, Wang X, He X, Huang X, Chua TS (2020) MGAT: multimodal graph attention network for recommendation. *Inform. Process. Manag.* 57(5):102277.

[75] Liu Y, Yang S, Lei C, Wang G, Tang H, Zhang J, Sun A, Miao C (2021) Pre-training graph transformer with multimodal side information for recommendation. *Proc. of the 29th ACM Int. Conf. on Multimedia*, Virtual Event China, 2021, 2853–2861.

[76] Mousselly-Sergieh H, Botschen T, Gurevych I, Roth S (2018) A multimodal translation-based approach for knowledge graph representation learning. *Proc 7th Joint Conference on Lexical and Computational Semantics*, New Orleans Louisiana USA, 2018, 225–234.

[77] Sun R, Cao X, Zhao Y, Wan J, Zhou K, Zhang F, Wang Z, Zheng K (2020) Multi-modal knowledge graphs for recommender systems. *Proc. 29th ACM Int. Conf. on Information*

and Knowledge Management, Virtual Event Ireland, 2020, 1405–1414.

[78] Pezeshkpour P, Chen L, Singh S (2018) Embedding multimodal relational data for knowledge base completion. *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels Belgium, 2018, 3208–3218.

[79] Song X, Feng F, Liu J, Li Z, Nie L, Ma J (2017) NeuroStylist: neural compatibility modeling for clothing matching. *Proc. 25th ACM international conference on Multimedia.* Mountain View California USA, 2017, 753–761.

[80] Pan W (2016) A survey of transfer learning for collaborative recommendation with auxiliary data. *Neurocomputing* 177(12):447–453.

[81] Iman M, Rasheed K, Arabnia HR (2022) A review of deep transfer learning and recent advancements. arXiv:2201.09679.

[82] Man T, Shen H, Jin X, Cheng X (2017) Cross-domain recommendation: an embedding and mapping approach. *Proc. 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne Australia, 2017, 2464–2470.

[83] Zhu Y, Ge K, Zhuang F, Xie R, Xi D, Zhang X, Lin L, He Q (2021) Transfer-meta framework for cross-domain recommendation to cold-start users. *Proc. 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Virtual Event Canada, 2021, 1813–1817.

[84] Zhu Y, Tang Z, Liu Y, Zhuang F, Xie R, Zhang X, Lin L, He Q (2022) Personalized transfer of user preferences for cross-domain recommendation. *Proc. 15th ACM Int. Conf. on Web Search and Data Mining*, Tempe AZ USA, 2022, 1507–1515.

[85] Wang L, Guo B, Yang Q (2018) Smart city development with urban transfer learning.

Computer 51(12):32–41.

[86] Wei Y, Zheng Y, Yang Q (2016) Transfer knowledge between cities. *Proc 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, San Francisco CA USA, 2016, 1905–1914.

[87] Chen Y, Wang X, Fan M, Huang J, Yang S, Zhu W (2021) Curriculum meta-learning for next POI recommendation. *Proc. 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Virtual Event Singapore, 2021, 2692–2702.

[88] Farseev A, Nie L, Akbari M, Chua TS (2015) Harvesting multiple sources for user profile learning a big data study. *Proc 5th ACM on Int. Conf. on Multimedia Retrieval*, Shanghai China, 2015, 235–242.

[89] Farseev A, Samborskii I, Filchenkov A, Chua TS (2017) Cross-domain recommendation via clustering on multi-layer graphs. *Proc. 40th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Shinjuku Tokyo Japan, 2017, 195–204.

[90] Wang X, He X, Nie L, Chua TS (2017) Item silk road: recommending items from information domains to social users. *Proc. 40th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Shinjuku Tokyo Japan, 2017, 185–194.

[91] Gao C, Huang C, Yu Y, Wang H, Li Y, Jin D (2019) Privacy-preserving cross-domain location recommendation. *IMWUT* 3(1):1–21.

[92] Li B, Yang Q, Xue X (2009) Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. *Proc. 21st Int. Joint Conf. on Artificial Intelligence*, Pasadena California USA, 2009, 2052–2057.

[93] Xin H, Lu X, Xu T, Liu H, Gu J, Dou D, Xiong H (2021) Out-of-town recommendation with

travel intention modeling. *Proc. 35th AAAI Conf. on Artificial Intelligence*, Virtual Event, 2021, 35(5):4529–4536.