Modeling Travel Behavior Similarity with Trajectory Embedding

Wenyan Yang[1], Yan Zhao[1], Bolong Zheng[2,3(✉)], Guanfeng Liu[1], and Kai Zheng[4]

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
20164227005@stu.suda.edu.cn, {zhaoyan,gfliu}@suda.edu.cn
[2] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
zhengblong@mail.sysu.edu.cn
[3] Aalborg University, Aalborg, Denmark
[4] Big Data Research Center, University of Electronic Science and Technology of China, Chengdu, China
zhengkai@uestc.edu.cn

Abstract. The prevalence of GPS-enabled devices and wireless communication technologies has led to myriads of spatial trajectories describing the movement history of moving objects. While a substantial research effort has been undertaken on the spatio-temporal features of trajectory data, recent years have witnessed the flourish of location-based web applications (i.e., Foursquare, Facebook), enriching the traditional trajectory data by associating locations with activity information, called activity trajectory. These trajectory data contain a wealth of activity information and offer unprecedented opportunities for heightening our understanding about human behaviors. In this paper, we propose a novel framework, called TEH (Trajectory Embedding and Hashing), to mine the similarity among users based on their activity trajectories. Such user similarity is of great importance for individuals to effectively retrieve the information with high relevance. With the time being separated into several slots according to the activity-based temporal distribution, we utilize trajectory embedding technique to mine the sequence property of the activity trajectories by treating them as paragraphs. Then a hash-based method is presented to reduce the dimensions for improving the efficiency of users' similarity calculation. Finally, extensive experiments on a real activity trajectory dataset demonstrate the effectiveness and efficiency of the proposed methods.

Keywords: Activity trajectory · User similarity
Trajectory embedding

1 Introduction

The increased popularity of various GPS-equipped smart devices, such as smart phones, personal navigation devices, and on-board diagnostics, has resulted in

J. Pei et al. (Eds.): DASFAA 2018, LNCS 10827, pp. 630–646, 2018.
https://doi.org/10.1007/978-3-319-91452-7_41

a huge volume of spatial trajectories (e.g., the GPS trajectories of vehicles), each of which consists of a sequence of time-ordered spatial points. This inspires massive efforts on analyzing the huge scale trajectory data from various aspects in the literature, including trajectory clustering [1,2], and knowledges/patterns mining from trajectory data [3–5], to name a few.

These studies mainly put the focus on the spatio-temporal features on the trajectory data. However, recent years have witnessed a revolution in location-based social network (LBSN) services, such as Foursquare[1], Facebook[2], and Flicker[3], in which large amounts of users' trajectories associated with activity information have been accumulated. These trajectories, namely activity trajectories [6,7], contain footprints (or check-ins) which offers the data about where and when a user has been as well as what she has done. For instance, at Foursquare, users can check in the places where they are visiting and leave tips about what they are doing. Intuitively, an activity trajectory of a user can be formed, which is a finite sequence of time-stamped locations. Each of the activity trajectory is associated with a keyword to describe a specific activity performed at this location. For example, T_{r1}, T_{r2} and T_{r3} in Fig. 1 exemplify the activity trajectories, where each point is attached with a keyword to illustrate an activity performed by the users. Note that how to extract and classify the activities is orthogonal to the approaches in our work, so each activity is just regarded as a unique entry of a pre-defined activity vocabulary. With these trajectory data, people intend to know more information about behavior preferences and daily habits [2,8].

In order to solve this problem, some existing studies have been proposed to learn user-specific activities [9,10] and mine knowledge [11–13] from the individual spatial data and trajectory information. However, these works do not explore the correlation between users, which is of great significance to effectively retrieve information matching user' tastes and recommend new friends sharing similar behaviors with them. Then in the literature, Lv et al. [8] mine the similarity between users based on the routine activity, which is extracted from the raw trajectory data. Taking the semantics of visited positions into account, Chen et al. [14] propose some basic principles to measure user similarity based on trajectory patterns. Li et al. [15] extract latent topics of users' check-ins by using topic modelling and then adopt these topics to measure user similarity. However, these existing works concentrate on analyzing the geographic and semantic feature of locations to learn the users' similarity, but neglect temporal features, which is significant in the similarity measurement.

In this paper, we focus on mining similar users based on their time-ordered activity trajectories regardless of geographic feature, and proposing a framework, referred to as Trajectory Embedding and Hashing (TEH), to model users' activity trajectory histories and to explore the similarity between users. The main idea of our approach is that, observing that performing the same activity in different time may refer to different lifestyles, we utilize Gaussian Mixture

[1] http://www.foursquare.com.
[2] http://www.facebook.com.
[3] http://www.flickr.com.

Model (GMM) [16] widely used in density estimation to model the temporal distribution of activities and apply a time-partition procedure to divide time into independent segments. A trajectory embedding algorithm is then developed to mine the sequence property of the activity trajectories by treating them as paragraphs. Finally, a hashing method is presented for the purpose of dimensionality reduction.

Our main contributions are summarized as follows:

1. We mine the similarity between users based on both their activity information and the temporal distribution feature of these activities embedded in the trajectories.
2. We utilize Gaussian Mixture Model to fit the temporal distribution of activities, which can achieve good effect, and then divide time into segments based on this distribution.
3. To the best of our knowledge, this is the first work that applies paragraph vector model, a widely used natural language processing technique, to map the activity trajectories and users into feature vectors, and then adopt a hash-based similarity calculation method to find similar users.
4. We conduct extensive experiments on real trajectory dataset, which empirically demonstrate the good efficiency and effectiveness of our proposed model.

The rest of this paper is organized as follows. Section 2 surveys the related work under different problem settings. Section 3 introduces the preliminary concepts and overviews our model. The proposed algorithms and related techniques are presented in Sect. 4, followed by the empirical study in Sect. 5. Finally, Sect. 6 concludes this paper and outlines the directions to further extend our work.

2 Related Work

In this section, we discuss prior work related to user similarity mining approaches and Neural Embedding Representation, which is the core technology in our work.

2.1 User Similarity

With the rapid development of positioning technologies and GPS devices, it is possible to record the movement trajectories of users in time. This attracts increasing attention of researchers, followed by a large number of trajectory research works [17–20]. Zheng et al. [21] propose spatial query processing for fuzzy objects. Shang et al. [22] study a novel problem of planning unobstructed paths in traffic-aware spatial networks. Zheng et al. [23] studies approximate keyword search in semantic trajectory.

Most of the above work tend to attach great importance to trajectory data itself by ignoring the similarity of moving objects. Recently, taking both the sequence property of user behaviors and the hierarchy property of geographic locations into account, Li et al. [24] develop a framework to model users' location histories and measure the similarity among them. A two-stage approach is

proposed by Lv [8] to mine users' long-term activity similarity based on their trajectories. They first introduce routine activities to capture users' long-term activity regularities and then calculate user similarity hierarchically based on the routine activities. Lee and Chung [25] propose a method to calculate the user similarity on the basis of the semantics of the location. However, these approaches mine the user similarity only based on the spatial and semantical features without considering the time distribution of activities in locations. Our work is to find behaviorally similar users based on the temporal distribution feature of activities embedded in all check-in points of the trajectories.

2.2 Neural Embedding Representation

Neural embedding representations are widely used in Natural Language Processing (NLP) [26,27]. Mikolov et al. [28] propose two novel model architecture - continuous Bags-of-Words and Skip-gram for learning continuous vector representations of words from large datasets. These distributed representations can capture word similarities in a semantic level and have good compositionality. Paragraph vector is introduced on the basis of word vector by Le and Mikolov [29], which projects words and documents into a single semantic space. It compensates for some key weaknesses of bag-of-words models, which lose the ordering of words and ignore their semantic information.

Meanwhile, these distributed representation methods are applied to the work in other fields successfully. Tajima et al. [30] model user activities on the Web with paragraph vectors. They consider users and web activities as paragraphs and words respectively. The distributed representations are used among the user-related prediction tasks. Zhao et al. [31] utilize paragraph vectors on the traditional medical system for similar cases recommendation. They try to utilize the semantic representation of sentences in a continuous space to understand the cases. Zhang et al. [32] use multi-modal embedding to achieve online local event detection in geo-tagged tweet streams. They capture short-text semantics by learning embedding of the location, time and text, and then perform online clustering by using a novel Bayesian mixture model. In this paper, we will capture the potential behavior features of users via trajectory embedding technology.

3 Preliminaries

In this section, we present some preliminary concepts and give an overview of the proposed model. Table 1 lists the major notations used throughout the paper.

3.1 Preliminary Concepts

Definition 1 (Activity). *An activity, denoted by α, is a type of action that a user can take at some place of interest (i.e., dining, sport and entertaining). We use $\mathbb{A}$ to denote the union of all the activities, which can be performed by the users.*

Table 1. Summary of notations

Notation	Definition
α	Activity
$\mathbb{A}$	Activity set
p	Check-in point
$p.l$	Location of check-in point p
$p.t$	Time stamp of check-in point p
$p.\alpha$	Activity of check-in point p
T_r	Activity trajectory
θ	Time partition ratio
U_i	User vector
W_i	Check-in vector
b_i	A binary code
k	Binary code length

Definition 2 (Check-in Point). *A check-in point, denoted by* $p = <p.l, p.t, p.\alpha>$*, is a time-stamped location point associated with an activity* $p.\alpha$*, where* $p.l : (x, y)$ *stands for the longitude and latitude information of the location at time stamp* $p.t$*.*

Definition 3 (Activity Trajectory). *An activity trajectory, denoted by* T_r*, is a finite sequence of check-in points, i.e.,* $T_r = (p_1, p_2, \ldots, p_n)$*.*

An activity trajectory T_r is a sequence of historical records representing not only where and when a user has been but also what she has done. Different users may exhibit similarity with respect to their traversed activity trajectories, and therefore two users can be correlated based on their activity trajectory similarity. In the rest of the paper, we will use *trajectory* and *activity trajectory* interchangeably when the context is clear.

For instance, Fig. 1 shows an example of several users $U = \{u_1, u_2, u_3\}$ and all activities $\mathbb{A}$ along the routes of the users. Each user has an activity trajectory, in which the spatio-temporal data and the information about her activities are embedded.

3.2 The Framework of TEH

Figure 2 shows the overview of the proposed framework, which basically comprised three parts: time partition, trajectory embedding and hash-based similarity calculation. In this work, we first employ the Gaussian mixture model to obtain the temporal distribution of activities. Then we divide the whole day into multiple segments based on the temporal distribution. In the second step, a trajectory embedding algorithm, which can capture the sequence property of

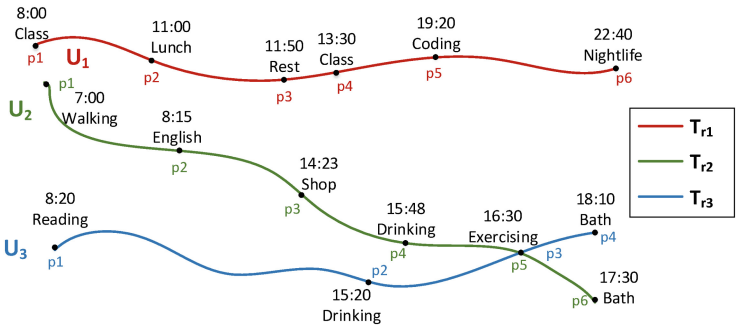


Fig. 1. Running example

the activity trajectories, is designed to map users' historical activity trajectories and users into vectors for quantifying these data. Due to the high computational cost of the high dimensional vectors, we introduce a hash-based user similarity calculation method for dimensionality reduction by mapping these vectors into compact binary codes, which can improve computational efficiency. We will detail the discussion for each part in the next section.

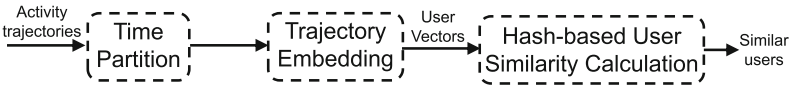


Fig. 2. TEH framework overview.

4 TEH Model

In this section, we detail the process of the proposed TEH model, including distribution-based time partition, trajectory embedding and hash-based user similarity calculation.

4.1 Distribution-Based Time Partition

Temporal Distribution of Activity. We observe that people performing the same activity in different time slots may have different behavior characteristics. In other words, two persons who perform a same activity in the same time slot can be considered as similar users, which play an important role in recommendation applications (i.e., potential friend recommendation, product recommendation). In this subsection, we design a method of time slot partition in terms of temporal distribution of all activities. We first introduce the Gaussian mixture model to model the temporal distribution of all activities, and then divide the whole day into time slots.

A Gaussian mixture model, widely used in density estimation, is employed to fit the temporal distribution in terms of the number of users performing the activity. Specifically, given an activity α, we denote the number of users performing this activity α at the time unit t as y_t. The Gaussian mixture model can be expressed as Eq. (1), assuming that the component $N(y_t, \mu_k, \sigma_k^2)$ obeys the single-Gaussian distribution.

$$p(y_t) = \sum_{k=1}^{K} \omega_k N(y_t, \mu_k, \sigma_k^2) \tag{1}$$

$$s.t. \sum_{i=1}^{n} \omega_k = 1, 0 \leq \omega_k \leq 1.$$

where K is the number of mixture components, and ω_k is the corresponding weight of the k-th component. μ_k and σ_k^2 represent the mean and covariance of the k-th component.

The parameters of $\omega_k, \mu_k, \sigma_k^2$ can be estimated by Expectation-Maximization (EM) algorithm, which is implemented by the repeating iterations between Expectation step and Maximization step as follows:

1. Initialize all parameters randomly. One day will be split into 24 units and each hour represents a unit.
2. In the E-step, the posterior probability of observation t belonging to k-th component is given as follow:

$$\gamma_{tk} = \frac{w_k N(y_t, \mu_k, \sigma_k^2)}{\sum_{k=1}^{K} w_k N(y_t, \mu_k, \sigma_k^2)}, t = 1, 2, \ldots, T; k = 1, 2, \ldots, K \tag{2}$$

   where t is the time unit when user conducts the activity.
3. In the M-step, the parameters will be derived under new posterior probabilities. The updated parameters are computed by Eq. 3.

$$\begin{aligned} \mu_k &= \frac{\sum_{t=1}^{T} \gamma_{tk} y_t}{\sum_{t=1}^{T} \gamma_{tk}}, k = 1, 2, \ldots K \\ \sigma_k^2 &= \frac{\sum_{t=1}^{T} \gamma_{tk} (y_t - \mu_k)^2}{\sum_{t=1}^{T} \gamma_{tk}}, k = 1, 2, \ldots, K \\ \omega_k &= \frac{\sum_{t=1}^{T} \gamma_{tk}}{T}, k = 1, 2, \ldots K \end{aligned} \tag{3}$$

4. Repeat steps 2 and 3 until converge.

As a result, we can get the distribution of all the activities. Figure 3 depicts the probability density distribution of the activity - Nightlife. It can be clustered as a mix of two Gaussian distributions. For each activity, by observing the change of number of people performing this activity with time, we will choose the K value based on the number of peak points and obtain the distribution of the

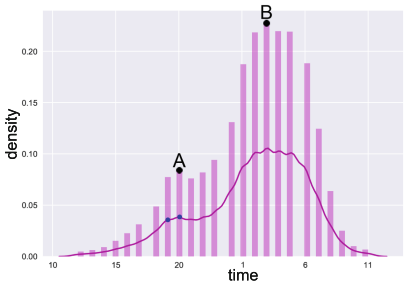


Fig. 3. Activity time distribution of Nightlife.

activity. For instance, Fig. 3 shows the time distribution of nightlife, in which the number of peak points (i.e. A and B) determines the K value (i.e., $K = 2$).

Time Partition. Based on the distribution of the activity, the comparative ratio of density in adjacent time unit, which is usually defined as the ratio between the small density and large density of two adjoining time units, is utilized to partition time. A threshold θ is set to partition time. Specifically, if the ratio of density is greater than or equal to the threshold, the two time units can be merged into a same time slot. Take Fig. 3 as an example, the comparative density ratio between unit 19 and 20 is 0.872. In the same time slot, users participating in the same activity can be considered to have similar behavior. Time partition will cluster crowds with similar density together for each activity.

4.2 Trajectory Embedding

As mentioned earlier, an activity trajectory is a finite sequence of check-in points, each of which is associated with a specific activity. To measure the similarity between users, we take into account not only the activities involved in the trajectories, but also the sequence of the activities. Hence, we employ a trajectory embedding algorithm that maps users' trajectories and users into vectors in order to capture the sequence property of these check-ins.

The trajectory embedding algorithm summarizes the sequence of each user's activities using the Paragraph Vector [29], which is an unsupervised approach that learns continuous distributed vector representations from pieces of text. In specific, we consider activities with corresponding time as words, and users as paragraphs, each of which has a main idea that represents the topic of the whole paragraph. In our work, this topic refers to behavior preference of each user. Thus, people having the similar topics would share similar preferences. The Paragraph vector model adds a vector to extract the topic based on the word embedding, which is the user vector what we need to compute the similarity between users.

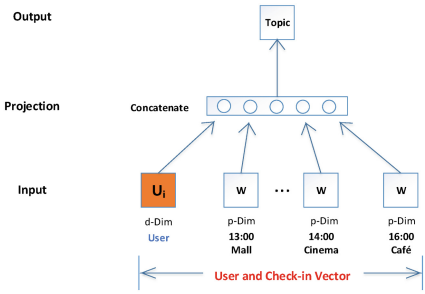


Fig. 4. The architecture of embedding.

Figure 4 demonstrates the architecture of trajectory embedding. In this model, the concatenation of user vector with a context of some check-in points is used to conclude the topic. The user vector represents the missing information from the current context and can act as a memory of the topic of the activity trajectory.

After mapping each user into a unique vector that is represented by a column in matrix U and each check-in into a unique vector that is represented by a column in matrix W, we apply the Paragraph Vector model to sequences of user trajectories. Given a sequence of check-ins $(c_{i,1}, c_{i,2}, \ldots, c_{i,N_i})$ generated by a user u_i and N_i is the size of this sequence, the objective of the vector model for this sequence is to maximize the average log probability:

$$\frac{1}{N_i} \sum_{n=j}^{N_i - j} logp(c_{i,k} | c_{i,k-j}, \ldots, c_{i,k+j}, u_i). \tag{4}$$

where $(c_{i,k-j}, \ldots, c_{i,k+j})$ is a sliding window, $c_{i,k}$ is the target check-in and other check-ins are contexts. The probability can be calculated using the softmax function according to PV_DM [29] as follows:

$$p(c_{i,k} | c_{i,k-j}, \ldots, c_{i,k+j}, u_i) = \frac{exp(\omega_{c_{i,k}}^{\mathrm{T}} \upsilon_i)}{\sum_{c \in |C|} exp(\omega_c^{\mathrm{T}} \upsilon_i)}. \tag{5}$$

where C is the union of all the possible check-ins, $\omega_{c_{i,k}}$ is the output vector of check-in $c_{i,k}$, and υ_i is the concatenated input vector of the context check-in vectors $\upsilon_{c_{i,k-j}}, \ldots, \upsilon_{c_{i,k+j}}$ and user vector U_i, that is $\upsilon_i = [\upsilon_{c_{i,k-j}}^{\mathrm{T}}, \ldots, \upsilon_{c_{i,k+j}}^{\mathrm{T}}, U_i]^{\mathrm{T}}$. The user vector U_i can be viewed as a feature extraction of user behavior intentions and is what we need to compare the similarity between users.

The user vectors and check-in vectors are usually trained with Stochastic Gradient Descent (SGD), in which the gradient is obtained via backpropagation. In [29], Le and Mikolov develop hierarchical softmax based on a binary Huffman tree structure for fast training. In this paper, we employ negative sampling method [33] for optimization instead of hierarchical softmax. We randomly choose a check-in point and K negative instances at each time segment. Then

we minimize the following objective to log probability for the sampled check-in points:

$$J_c = -log\sigma(\omega_{c_{i,k}}^{\mathrm{T}} v_i) - \sum_{j=1}^{J} log\sigma(-\omega_j^{\mathrm{T}} v_i). \tag{6}$$

where $\sigma(x) = 1/(1 + exp(-x))$ is a sigmoid function. Then this objective can be trained using SGD. As the updating rules for parameters is easily derived, we omit the detail in this paper.

Take Fig. 1 as an example, we can obtain the vector representations of the three users through trajectory embedding model. Specifically, if we set the size of vectors and sliding windows to 5 and 3 respectively, we can get three 5-dimensional user vectors.

4.3 Hash-Based User Similarity Calculation

The main computational challenge lies in highly time-consuming similarity calculation for all the user vectors with high dimensions. Hence, we employ a dimensionality reduction method, spherical hashing algorithm, to transform the user vectors into the corresponding binary codes, and then compute the similarity between users based on spherical Hamming distance.

Binary Code Learning. In this section, we utilize spherical hashing algorithm based on hyperspheres to learn the compact binary codes from user vectors.

Given a set of user vectors in a D-dimensional space, $\mathbb{U} = \{u_1, u_2, \ldots, u_n\}$, $u_i \in \mathbb{R}^d$, a binary code corresponding to a user vector u_i is expressed by $b_i = \{0, 1\}^k$, where k is the length of the code. Since each binary code has k bits, we need k different hash functions to map user vectors. Each hash function, $h_m(u)(1 \leq m \leq k)$, is defined with a pivot $p_m \in \mathbb{R}^d$ and a radius $t_m \in \mathbb{R}^+$ as following:

$$h_m(u) = \begin{cases} 0, & \text{if} \quad d(p_m, u) > t_m \\ 1, & \text{if} \quad d(p_m, u) \leq t_m \end{cases} \tag{7}$$

where $d(x, y)$ denotes the Euclidean distance between vector x and y. The hash function determines whether the vector u is inside the hypersphere whose center is p_m and radius is t_m, where p_m and t_m for the k different hyperspheres can be obtained by an iterative optimization process [34]. To save space, the procedure of calculating p_m and t_m is omitted here. Through multiple hash functions, we can obtain the binary code for each user vector.

Note that the multiple hash function construction must follow two criteria, balanced partitioning of data space for each hash function and the independence between hash functions. Balanced partitioning indicates that the user vectors have the equal probability to be divided inside and outside the given hypersphere, so we define each hash function h_m to have the equal probability for $+1$ and -1 bits respectively as following:

$$Pr[h_m(u) = 1] = \frac{1}{2}, \quad u \in \mathbb{U}, 1 \leq m \leq c \tag{8}$$

A probabilistic event E_m is defined to represent the case of $h_m(u) = 1$. As we all know, two events E_i and E_j are independent if and only if $Pr[E_i, E_j] = Pr[E_i] \cdot [E_j]$. After the balanced partitioning process of user vectors for each bit (Eq. (8)), the independence between two bits should satisfy the following equation:

$$\begin{aligned} &Pr[h_i(u) = 1], h_j(u) = 1] \\ &= Pr[h_i(u) = 1] \cdot Pr[h_j(u) = 1] \\ &= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \end{aligned} \tag{9}$$

where $u \in \mathbb{U}, 1 \leq i, j \leq c$.

User Similarity Calculation. Next we explain how to measure the similarity between two users based on their binary codes. As mentioned before, the user vectors, reflecting the users' behavior characteristics, are mapped into the corresponding binary codes. The two users are more similar with each other if they have more similar binary codes. In this part, we apply the spherical Hamming distance [34], $d_{shd}(b_i, b_j)$, to measure the similarity between two binary codes to find the similar users. $d_{shd}(b_i, b_j)$ is defined as follows:

$$d_{shd}(b_i, b_j) = \frac{|b_i \oplus b_j|}{|b_i \wedge b_j|}. \tag{10}$$

where $b_i \oplus b_j$ is the XOR bit operation and $b_i \wedge b_j$ is the AND bit operation. $|\cdot|$ denotes the number of $+1$ bits in the binary code. Two users are more similar in behaviors with the smaller distance between them.

5 Experiment

In this section, we conduct extensive experiments to validate the effectiveness and efficiency of our model. All the algorithms are implemented on an Intel Core i5-6200U CPU @ 2.40 GHZ with 8 GB RAM.

5.1 Experimental Setup

Datasets. We use a real trajectory dataset generated in Foursquare by 2070 users who live in California, USA. Each record is associated with a check-in point of a user, which contains user id, venue name and category, locations and timestamp. There are 483813 check-in records in this dataset.

Parameters. Table 2 lists all the parameters we used throughout the experiments, that all the parameters are assigned the default values unless specified explicitly.

Table 2. Default values of parameters

Parameter	Default value	Description
θ	0.8	Time partition ratio
d	300	User vector dimension
c	64	Binary code length

Baselines. We compare our approach with the two baselines. The first is Jaccard index [35], which is a popular method to measure the similarity. The Jaccard similarity between two users, (i.e., u and v), can be computed as follows:

$$sim_j(u,v) = \frac{T_{r,u} \bigcap T_{r,v}}{T_{r,u} \bigcup T_{r,v}} \tag{11}$$

where $T_{r,u}$ is the activity trajectory of user u.

The second baseline is to compute the Pearson coefficient between users to measure the users similarity. In specific, each user is denoted by a matrix $M_{m\times n}$, where each element $u_{i,j}$ represents the probability that the user conducts the activity i at time unit j. The Pearson coefficient between user u and user v can be computed as follows:

$$sim_p(u,v) = \frac{\sum_m \sum_n (U_{mn} - \overline{U})(V_{mn} - \overline{V})}{\sqrt{(\sum_m \sum_n (U_{mn} - \overline{U})^2)(\sum_m \sum_n (V_{mn} - \overline{V})^2)}} \tag{12}$$

Evaluation Method. A user study method is employed to evaluate the performance of our approaches and the two baselines, in which 100 volunteers are required to vote for the most similar user for each user. For each experiment, we run 200 test cases, which are randomly chosen from the dataset. We use the voting results to evaluate the effectiveness of all the methods.

5.2 Experimental Results

In this subsection, we will show the effectiveness and efficiency of our proposed model based on user study results. We declare some notations shown in the following figures. TEH represents the similarity measure of our proposed model. Trajectory embedding and cosine similarity (TEC) is our model without hashing step and use cosine similarity to compare users. Jaccard and Pearson denote the two baseline methods.

Effectiveness of TEH. Through user study, we obtain the score of three methods. According to the score, we could judge the superiority of our method. The user voting results are listed below.

Table 3. Effect of θ

θ	0.7	0.75	0.8	0.85	0.9
TEH	110	118	124	120	115
$Jaccard$	52	45	41	46	48
$Pearson$	38	37	35	34	37

Effect of θ. The votes of three methods are shown in Table 3. It can be seen from the table that the performance increases originally. With lower ratio, more people can be mistaken as similar users who are not in fact. Then, as θ continues to increases, the votes of our model will decline. A higher ratio means more subtle time partition, which will split the relationship between users. Besides, the votes of our model are far more than that of baselines, which directly shows that our model is better than the baselines.

Table 4. Effect of d

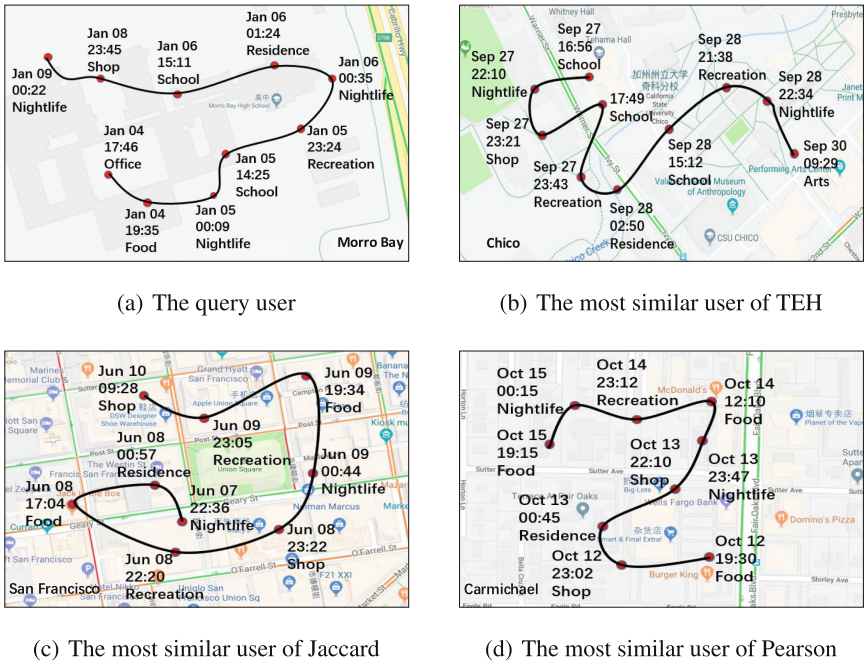
d	200	300	400	500
TEH	115	124	126	127
$Jaccard$	46	41	40	39
$Pearson$	39	35	34	34

Effect of d. The voting result of all the methods are presented with the change of user vector dimension d in Table 4. We can observe, for our model, the votes increase as d grows. However, we also notice that the increase becomes slower when $d \geq 300$ since with higher vector dimensions there is increasing chance that subsequently added dimensions are redundant.

Table 5. Effect of k

k	32	64	128	256
TEH	116	124	125	126
$Jaccard$	48	41	40	38
$Pearson$	36	35	35	36

Effect of k. In Table 5, it depicts the votes of three methods changing over binary code length. As we can see, at the beginning the performance improves as k increases. Moreover, we observe that there is little improvement on votes after k exceeds 64. Because binary code length reaches a certain value, the relationship between users can be well preserved.



(a) The query user

(b) The most similar user of TEH

(c) The most similar user of Jaccard

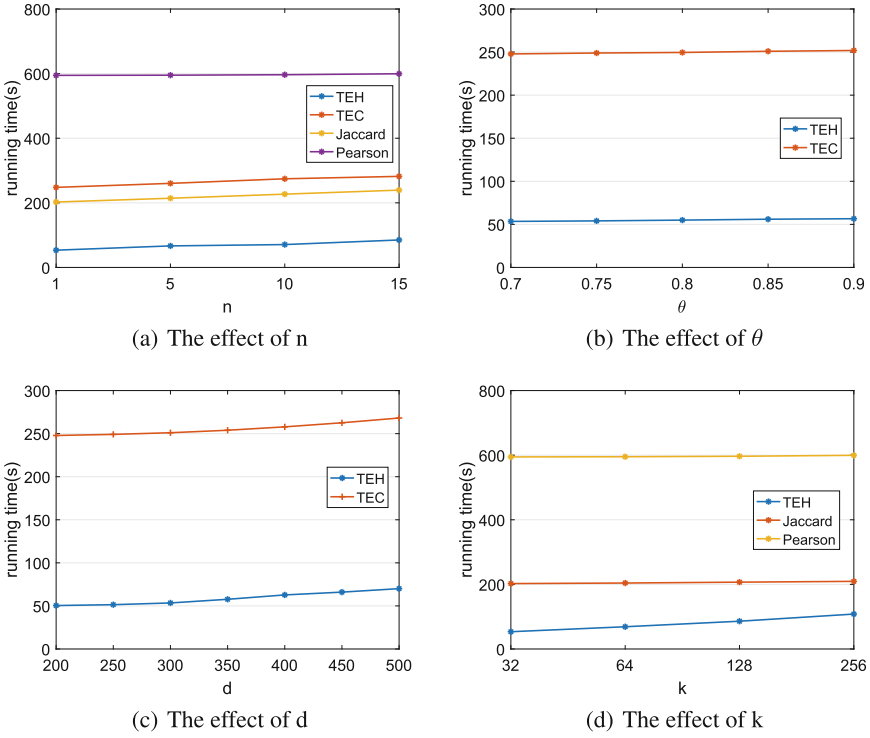
(d) The most similar user of Pearson

Fig. 5. The query user and the most similar user of three methods.

Figure 5 shows that, given a query user, our model can find out more similar users than baselines. From the picture, we find that the query user often stay at school in the afternoon and have colorful nightlife. Besides, she sleeps late. The two baselines discover the most similar users for the query user, just considering the users' nightlife activity, while ignoring the learning at school in the afternoon. Our model can find a user, whose behavior is more similar to that of the query user, i.e., learning at school in the afternoon and staying up late. Our model can exactly capture the sequence property of user trajectory, while baselines only take the proportion of user activities into account. The baselines usually just consider the major activity and leave the consideration of the remaining activity.

Efficiency of TEH. In this part, we vary the values of parameters in Table 2 to compare our model with baselines and observe the effect of these parameters in efficiency. We use the total running time of finding the top-n similar user for each user to represent the efficiency.

Effect of n. Figure 6(a) shows the running time of different methods. We can perceive that TEH improves a lot than TEC and baselines in efficiency. When the number of users is large, calculating cosine similarity of user vectors is highly time-consuming. Therefore, it is particularly vital to operate hashing procedure. Besides, the running time increases a little as n increases. With n value increasing, we need to search more users who are similar to the query user.



(a) The effect of n (b) The effect of θ

(c) The effect of d (d) The effect of k

Fig. 6. The effect of parameters.

Effect of θ. In Fig. 6(b), we find that θ has little effect on the running time of our methods. This is because different θ values result in different time partition, however, it has little impact on the running time of trajectory embedding.

Effect of d. From Fig. 6(c), we can find that the efficiency will cut down as the user vector size increases. This is mainly because high vector dimension will lead to more time for calculating the similarity between users.

Effect of k. In Fig. 6(d), the overall time of TEH will increase drastically with k increasing, since it takes more time to train binary code. When the binary code length increases, TEH need to construct more hash functions, which will lead to a longer training time.

6 Conclusion

In this paper, we propose a novel framework, referred to as TEH, to model the users' activity trajectories and effectively mine the similarity between users. The core of our framework is the trajectory embedding technique and hash-based similarity measurement, which employ paragraph vector model to map the trajectories into vectors by treating them as paragraphs and then transform

the vectors into compact binary codes for efficient user similarity calculation. To our best knowledge, this is the first work applying the paragraph vector model, a widely used Natural Language Processing technique, to model the trajectory data as vectors. Through a series of experiments, we validate our proposal and demonstrate that the proposed TEH framework has excellent performance under various conditions. As for future work, we plan to design more sophisticated user similarity mining algorithm by taking the spatial information and users' social relationships into account.

Acknowledgement. This research is partially supported by the Natural Science Foundation of China (Grant Nos. 61502324, 61532018).

References

1. Shang, S., Zheng, K., Jensen, C.S., Yang, B., Kalnis, P., Li, G., Wen, J.R.: Discovery of path nearby clusters in spatial networks. TKDE **27**(6), 1505–1518 (2015)
2. Hung, C.-C., Peng, W.-C., Lee, W.-C.: Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. VLDB J. **24**(2), 169–192 (2015)
3. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Trajectory similarity join in spatial networks. PVLDB **10**(11), 1178–1189 (2017)
4. Shang, S., Ding, R., Zheng, K., Jensen, C.S., Kalnis, P., Zhou, X.: Personalized trajectory matching in spatial networks. VLDB J. **23**(3), 449–468 (2014)
5. Su, H., Zheng, K., Zeng, K., Huang, J., Sadiq, S., Yuan, N.J., Zhou, X.: Making sense of trajectory data: a partition-and-summarization approach. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE), pp. 963–974. IEEE (2015)
6. Kai, Z., Zheng, B., Xu, J., Liu, G., An, L., Li, Z.: Popularity-aware spatial keyword search on activity trajectories. WWWJ **20**(4), 1–25 (2016)
7. Zheng, K., Shang, S., Yuan, N.J., Yang, Y.: Towards efficient search for activity trajectories. In: ICDE, pp. 230–241 (2013)
8. Lv, M., Chen, L., Chen, G.: Mining user similarity based on routine activities. Inf. Sci. **236**(1), 17–32 (2013)
9. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Building personal maps from GPS data. Ann. N. Y. Acad. Sci. **1093**(1), 249 (2006)
10. Liao, L., Fox, D., Kautz, H.: Learning and inferring transportation routines. In: AAAI, pp. 348–353 (2004)
11. Shang, S., Ding, R., Yuan, B., Xie, K., Zheng, K., Kalnis, P.: User oriented trajectory search for trip recommendation. In: EDBT, pp. 156–167 (2012)
12. Su, H., Zheng, K., Huang, J., Wang, H., Zhou, X.: Calibrating trajectory data for spatio-temporal similarity analysis. VLDB J. **24**(1), 93–116 (2015)
13. Zheng, K., Zheng, Y., Yuan, N.J., Shang, S., Zhou, X.: Online discovery of gathering patterns over trajectories. TKDE **26**(8), 1974–1988 (2014)
14. Chen, X., Lu, R., Ma, X., Pang, J.: Measuring user similarity with trajectory patterns: principles and new metrics. In: Chen, L., Jia, Y., Sellis, T., Liu, G. (eds.) APWeb 2014. LNCS, vol. 8709, pp. 437–448. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11116-2_38
15. Li, W., Jiang, J., Li, G.: Mining user similarity based on users trajectories. In: ICCCBDA, pp. 557–562 (2014)

16. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: CVPR, p. 2246 (1999)
17. Zheng, K., Su, H., Zheng, B., Shang, S., Xu, J., Liu, J., Zhou, X.: Interactive top-k spatial keyword queries. In: IEEE, pp. 423–434, May 2015
18. Zheng, B., Zheng, K., Xiao, X., Su, H., Yin, H., Zhou, X., Li, G.: Keyword-aware continuous KNN query on road networks. In: ICDE, pp. 871–882 (2016)
19. Zheng, B., Su, H., Hua, W., Zheng, K., Zhou, X., Li, G.: Efficient clue-based route search on road networks. TKDE **29**(9), 1846–1859 (2017)
20. Wang, H., Zheng, K., Xu, J., Zheng, B., Zhou, X., Sadiq, S.: SharkDB: an in-memory column-oriented trajectory storage. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, pp. 1409–1418. ACM (2014)
21. Zheng, K., Zhou, X., Fung, P.C., Xie, K.: Spatial query processing for fuzzy objects. VLDB J. **21**, 1–23 (2012)
22. Shang, S., Liu, J., Zheng, K., Lu, H., Pedersen, T.B., Wen, J.: Planning unobstructed paths in traffic-aware spatial networks. GeoInformatica **19**(4), 723–746 (2015)
23. Zheng, B., Yuan, N.J., Zheng, K., Xie, X., Sadiq, S., Zhou, X.: Approximate keyword search in semantic trajectory database. In: ICDE, pp. 975–986 (2015)
24. Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., Ma, W.Y.: Mining user similarity based on location history. In: ACM SIGSPATIAL, p. 34 (2008)
25. Lee, M.-J., Chung, C.-W.: A user similarity calculation based on the location for social network services. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011. LNCS, vol. 6587, pp. 38–52. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20149-3_5
26. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. Computer Science (2013)
27. Collobert, R., Weston, J., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(1), 2493–2537 (2011)
28. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Computer Science (2013)
29. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML 2014, vol. 32, pp. II-1188–II-1196 (2014)
30. Tagami, Y., Kobayashi, H., Ono, S., Tajima, A.: Modeling user activities on the web using paragraph vector. In: International Conference on World Wide Web, pp. 125–126 (2015)
31. Zhao, Y., Wang, J., Wang, F.Y., Shi, X., Lv, Y.: Paragraph vector based retrieval model for similar cases recommendation. In: Intelligent Control and Automation, pp. 2220–2225 (2016)
32. Zhang, C., Liu, L., Lei, D., Yuan, Q., Zhuang, H., Hanratty, T., Han, J.: TrioVecEvent: embedding-based online local event detection in geo-tagged tweet streams. In: ACM SIGKDD, pp. 595–604 (2017)
33. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
34. Heo, J.P., Lee, Y., He, J., Chang, S.F.: Spherical hashing. In: CVPR, pp. 2957–2964 (2012)
35. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytol. **11**(2), 37–50 (2010)