Information Systems 54 (2015) 156–168

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Mining moving patterns for predicting next location

Meng Chen[a], Xiaohui Yu[a,b], Yang Liu[a,*]

[a] School of Computer Science and Technology, Shandong University, Jinan, Shandong 250101, China
[b] Department of Computer Science and Engineering, York University, Toronto, ON, Canada M3J 1P3

ARTICLE INFO

Article history:
Received 30 November 2014
Received in revised form
8 June 2015
Accepted 1 July 2015
Available online 10 July 2015

Keywords:
Moving patterns mining
Next location prediction
Time factor

ABSTRACT

Next location prediction has been an essential task for many location based applications such as targeted advertising. In this paper, we present three basic models to tackle the problem of predicting next locations: the Global Markov Model that uses all available trajectories to discover global behaviors, the Personal Markov Model that focuses on mining the individual patterns of each moving object, and the Regional Markov Model that clusters the trajectories to mine the similar movement patterns. The three models are integrated with linear regression in different ways. We then seek to further improve the accuracy of prediction by considering the time factor, with a focus on clustering the trajectories in different time periods, and present three methods to train the time-aware models to mine periodic patterns. Therefore, our proposed models have the following advantages: (1) we consider both individual and collective movement patterns in making prediction, (2) we consider the similarity between different trajectories, (3) we consider the time factor and build models that are suited to different time periods. We have conducted extensive experiments on a real dataset, and the results demonstrate the superiority of our proposed models over existing methods.

1. Introduction

With the wide spread use of positioning technology, it is increasingly possible to track the movement of people and other objects (e.g., vehicles), giving rise to a dazzling array of location-based applications. For example, GPS tracking using positioning devices installed on the vehicles is becoming a preferred method of taxi cab fleet management. The data collected by the location-based social applications (e.g., Foursquare) can also help with location-aware recommendations or advertising. As an example, let us say that Lily has just shared her current location through her favorite social application. If the area she will pass by is known in advance, it is possible to push plenty of information to her, such as the most popular restaurant and the products on promotions in that area.

Moreover, in an increasing number of cities, vehicles are photographed when they pass the surveillance cameras installed over highways and streets, and the vehicle passage records including the vehicle IDs, the locations of the cameras and the time are transmitted to the data center for storage and further processing. If we could predict the next locations of vehicles on the road, then we will be able to forecast the traffic conditions and recommend more reasonable routes to drivers to avoid or alleviate traffic jams.

In particular, both types of data contain three main attributes: the object, the location and the time-stamp. The availability of such spatial and temporal data gives rise to a large volume of applications, such as urban computing [1–3], finding popular routes [4,5], location-based recommendations [6,7], and route planning [8–12]. One key operation in such applications is next location prediction.

* Corresponding author.
E-mail addresses: chenmeng114@hotmail.com (M. Chen), xyu@sdu.edu.cn (X. Yu), yliu@sdu.edu.cn (Y. Liu).

http://dx.doi.org/10.1016/j.is.2015.07.001
0306-4379/© 2015 Elsevier Ltd. All rights reserved.

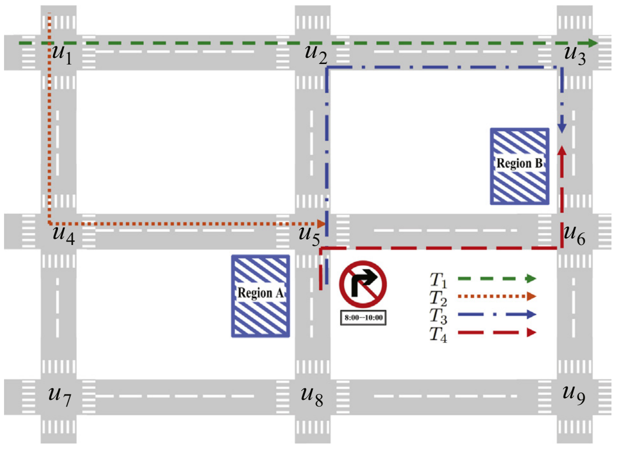

Fig. 1. An example of next location prediction.

The existing methods to next location prediction are to mine movement patterns with historical trajectories, and most of which fall into one of two categories: (1) methods that use only the historical trajectories of individual objects to discover individual movement patterns [13,14], and (2) methods that use the historical trajectories of all objects to identify collective movement patterns [15,16]. For both categories, the majority of the existing methods rely on mining frequent patterns and/or association rules to discover movement patterns for prediction. If the current trajectory sequence matches part of a movement pattern, we then make prediction using it. As shown in Fig. 1, there are four trajectories: $T_1 = \langle u_1, u_2, u_3 \rangle$, $T_2 = \langle u_1, u_4, u_5 \rangle$, $T_3 = \langle u_5, u_2, u_3 \rangle$, $T_4 = \langle u_5, u_6 \rangle$. Each trajectory is represented by a different type of line. If someone just drives from u_1 to u_4, since the trajectory sequence $\langle u_1, u_4 \rangle$ matches part of historical trajectory T_2, the next location of T_2 (i.e., u_5) will be the predicted next location of $\langle u_1, u_4 \rangle$. In practice, many historical trajectories contain this trajectory sequence $\langle u_1, u_4 \rangle$, and the most popular next location is returned.

However, the existing methods have a few drawbacks. First, prior studies on the location prediction have used the individual patterns or the collective patterns to predict the next locations, but very often the movements of objects are affected by both individual and collective properties. Second, similarities often exist between the trajectories. For example, the trajectory $T_1 = \langle u_1, u_2, u_3 \rangle$ is similar to $T_3 = \langle u_5, u_2, u_3 \rangle$, and they both have the same pattern $\langle u_2, u_3 \rangle$. Considering the similarity between trajectories may help mine the moving patterns more effectively and efficiently. Finally, the existing methods do not give proper consideration to the time factor. Different movement patterns exist in different time, for example, as shown in Fig. 1, during the non-rush hours, drivers who would like to go to Region A from Region B can choose the trajectory T_4. However, during the rush hours (e.g., 8:00–10:00), no turns are allowed at u_5, and drivers have to resort to a different (although a bit longer) trajectory T_3. Failing to take time factor into account would result in higher error rates in predicting the next locations.

In this paper, we propose three models to predict the next locations of moving objects given past trajectory sequences: the Global Markov Model (*GMM*) that uses all available trajectories to discover global behaviors of the moving objects based on the assumption that they often share similar movement patterns (e.g., people driving from A to B often take the same route), the Personal Markov Model (*PMM*) that focuses on modeling the individual patterns of each moving object using its own past trajectories, and the Regional Markov Model (*RMM*) that takes the similarity between trajectories into consideration and clusters the trajectories to mine the movement patterns. Each model can return the probabilities of the predicting next locations, and we combine them using linear regression in different ways to produce more complete and accurate predictors. Given a partial trajectory sequence, we can use the predictor obtain the probability of reaching each possible next location, and the top ranked ones are returned as answers.

In addition, the time factor also plays an important role in the next location prediction problem, and periodicity is an oft-occurring phenomenon in the movement of objects [17]. The movement patterns of objects vary from one time period to another (e.g., weekdays vs. weekends). Meanwhile, similarities also exist for different time periods (e.g., this Monday and next), and the movement patterns of moving objects tend to be cyclical. We consider three possible methods to partition the time dimension into smaller units so that more refined models can be built for each unit. We first propose a naive method, *Time Binning*, which partitions the time span into equi-sized time bins, and each trajectory is assigned to one of the time bins according to its time-stamp. A major drawback of this method is that its flexibility is limited, as the sizes of the time bins are fixed and equal. Therefore, we propose two improved methods: (1) *Distribution Clustering*, which clusters the time bins based on the similarity of the probability distributions of trajectories in them, and (2) *Hierarchical Clustering*, which merges the time bins to generate clusters hierarchy.

The performance of the proposed models is evaluated on a real dataset consisting of the vehicle passage records over a period of 31 days (1/1/2013–1/31/2013) in the City of Jinan. The experimental results confirm the superiority of the proposed methods over existing methods.

The contributions of this paper can be summarized as follows.

- We propose three models to predict the next location of a moving object: the *GMM* that uses all available trajectories to discover collective patterns, the *PMM* that models the individual patterns of each moving object using its own past trajectories, and the *RMM* that clusters the trajectories to mine the movement patterns. We combine the three models in different ways to obtain new models. To the best of our knowledge, the proposed models are the first ones that take a holistic approach and consider individual, collective movement patterns and the similarity between trajectories in making prediction.
- Based on the important observation that the movement patterns of moving objects often change over time, we propose methods that can capture the relationships between the movement patterns in different time periods, and use this knowledge to build more refined models that are better suited to different time periods.
- We conduct extensive experiments using a real dataset and the results demonstrate the effectiveness of the proposed models.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives the preliminaries of our work. Section 4 describes our approach of Markov modeling. Section 5 presents methods that take the time factor into consideration. The experimental results and performance analysis are presented in Section 6. Section 7 concludes this paper.

2. Related work

There have appeared a considerable body of work on knowledge discovery from trajectories, where a trajectory is defined as a sequence of locations ordered by timestamps. In what follows, we discuss two categories of studies that are most closely related to us.

2.1. Trajectory mining

Route planning: Several studies use GPS trajectories for route planning through constructing a complete route [18,4,19]. Chen et al. [18,4] search the *k* Best-Connected Trajectories from a database and discover the most popular route between two locations. Yuan et al. [19] find the practically fastest route to a destination at a given departure time using historical taxi trajectories.

Long-range prediction: Long-range prediction is studied in [20–23], where they try to predict the whole future trajectory of a moving object. Krumm [20] proposes a Simple Markov Model that uses previously traversed road segments to predict routes in the near future. Froehlich and Krumm [21] use previous GPS traces to make a long-range prediction of a vehicle's trajectory. Alvarez-Garcia et al. [22] present a system based on the generation of a Hidden Markov Model from the past GPS log and current location to predict the destination of a user when beginning a new trip. Simmons et al. [23] build a hidden Markov model (HMM) of the routes and destinations used by the driver with a low-cost GPS sensor and a map database.

Short-range prediction: Short-range prediction has been widely investigated [13–16], which is concerned with the prediction of only the next location. Some of these methods make prediction with only the individual movements [13,14], while others use the historical movements of all the moving objects [15,16]. Xue et al. [13] construct a Probabilistic Suffix Tree (PST) for each road using the taxi traces and propose a method based on Variable-order Markov Models (VMMs) for short-term route prediction. Jeung et al. [14] present a hybrid prediction model to predict the future locations of moving objects, which combine predefined motion functions using the object's recent movements with the movement patterns of the object. Monreale et al. [15] use the previous movements of all moving objects to build a T-pattern tree to make future location prediction. Morzy [16] uses a modified version of the PrefixSpan algorithm to discover frequent trajectories and movement rules with all the moving objects' locations.

All above works use historical trajectories to make location prediction, but there are some other studies which improve the accuracy of predicted locations with external information in addition to historical trajectories. For example, the length of trajectory, travel time, accident reports and driving habits have been incorporated into corresponding models to compute the probabilities of predicted locations [24–26]. Similarly, context information such as time-of-day, day-of-week, and velocity has been incorporated as the features in training the prediction model to help improve the accuracy [27,13]. However, since these studies mainly focus on utilizing external information, the proposed methods have little benefit in the absence of the aforementioned external information. Our work aims to solve a general next location prediction problem with three basic attributes (e.g. the id of moving object, location and time) that different kinds of trajectories always have. Therefore, the above studies which consider the external information are not applicable to our problem.

2.2. Social-media mining

In addition to the trajectories of moving objects, there has also appeared work on trajectory mining using social-media data [28,29,12]. The increasingly popular social-local-mobile applications have generated data such as geo-tagged photos and check-in locations, which can be regarded as sequences of visited locations. Zheng et al. [28] mine interesting locations and classical travel sequences in a given region with multiple users' GPS trajectories. Yin et al. [12] investigate the problem of trajectory pattern ranking and diversification based on geo-tagged social media. Hasan et al. [29] analyze urban human mobility and activity patterns using location-based data collected from social media applications.

Probably more related to the problem addressed in this paper is the work [30,31] on predicting next locations with semantic trajectories, where semantic trajectories are a sequence of visited places tagged with semantic information. Ye et al. [30] discover the underlying user movement pattern with the check-in category information. They first predict the category of user activities with a hidden Markov model and then predict the most likely location, given the estimated category distribution. Ying et al. [31] combine semantic information about the places with location data to improve the accuracy of predicting future locations. They first store the semantic trajectories in a tree-like structure and then choose the closest trajectory to make prediction by relying on the geographical and semantic features for the given trajectory sequence.

3. Preliminaries

In this section, we will explain a few terms that are required for the subsequent discussion, and define the problem addressed in this paper.

Definition 1 (*Sampling location*). For a given moving object o, it passes through a set of *sampling locations*, denoted by l, where each sampling location refers to a point or a region (in a two-dimensional area of interest) where the position of o is recorded.

For example, the positions of the cameras in the traffic surveillance system can be considered as the sampling locations. The sampling location hereinafter is also referred to as the location when the meaning is clear from the context.

Definition 2 (*Trajectory unit*). For a given moving object o, a *trajectory unit*, denoted by u, is the basic component of its trajectory. Each trajectory unit u can be represented by $(u.l, u.t)$, where $u.l$ is the id of the sampling location of the moving object at time-stamp $u.t$.

Definition 3 (*Trajectory*). For a moving object, its *trajectory* T is defined as a time-ordered sequence of trajectory units: $\langle u_1, u_2, \ldots, u_n \rangle$.

For a trajectory, the time difference between two neighbor trajectory units u_i and u_{i+1} should be less than a threshold (e.g., 1 hour), and the length of threshold is usually data-dependent. From Definition 2, T can also be represented as $\langle (u_1.l, u_1.t), (u_2.l, u_2.t), \ldots (u_n.l, u_n.t) \rangle$ where $u_i.t < u_{i+1}.t$ $(1 \le i \le n-1)$.

Definition 4 (*Candidate next locations*). For the sampling location l_i, we define a sampling location l_j as a *candidate next location* of l_i if a moving object can reach l_j from l_i without going through another sampling location first.

The set of candidate next locations can be obtained either by prior knowledge (e.g., locations of the surveillance cameras combined with the road network graph) or by induction from historical trajectories of moving objects.

Definition 5 (*Sampling location sequence*). For a trajectory $T = \langle (u_1.l, u_1.t), (u_2.l, u_2.t), \ldots, (u_n.l, u_n.t) \rangle$, its *sampling location sequence* T^l refers to a sequence of sampling locations appearing in the trajectory, denoted as $\langle u_1.l, u_2.l, \ldots, u_n.l \rangle$.

Definition 6 (*Prefix sequence*). For a sampling location l_i and a given sampling location sequence $T^l = \langle l_1, l_2, \ldots, l_n \rangle$, its *prefix sequence* L_i^j refers to a length-j subsequence of T^l ending with l_i.

Given such a trajectory sequence $T = \langle u_1, u_2, \ldots, u_n \rangle$, the *next location prediction* problem is to predict the location that the moving object will arrive at next.

4. Markov modeling

We choose to use Markov models to solve the next location prediction problem. Specifically, a state in the Markov model corresponds to a sampling location, and state transition corresponds to moving from one sampling location to the next.

In order to take into consideration both the collective, the individual movement patterns and the similarity between trajectories in making prediction, we propose three models: a Global Markov Model (*GMM*) to model the collective patterns, a Personal Markov Model (*PMM*) to model the individual patterns, and a Regional Markov Model (*RMM*) to infer some unseen patterns through modeling the similar trajectories. They are combined using linear regression to generate different predictors.

4.1. Global Markov model

The movements of objects often exhibit strong collective characteristics. For example, the trajectories of different moving objects traveling from one location A to another location B usually bear considerable similarities. One possible reason is the restrictions imposed by the road networks (e.g., only a limited number of possible routes from A to B). Another possible reason is that people tend to follow the crowd, and the majority of people usually prefer to take the well-beaten, typical path, especially when in an unfamiliar place. As a result, we decide to build a Global Markov Model (*GMM*) using the trajectories of all objects.

Using historical trajectories, we can train an order-N *GMM* to give a probabilistic prediction over the next sampling locations for a moving object, where N is a user-chosen parameter. For a given sampling location sequence $T = \langle l_1, l_2, \ldots, l_n \rangle$, let $p(l_{n+1}|T)$ be the probability that the object will arrive at location l_{n+1} next. Then the location l_{n+1} is given by

$$\begin{aligned} l_{n+1} &= \arg\max_{l \in \mathcal{L}} \{ p(l_{n+1} = l | T) \} \\ &= \arg\max_{l \in \mathcal{L}} \left\{ p\left(l_{n+1} = l | L_n^N \right) \right\}, \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the set of all locations. Essentially, this approach for each location l computes its probability of next visit, and selects the one that has the highest probability. The order-N *GMM* implies that the probability of arriving at a location l is independent of all but the immediately preceding N locations.

In order to use the order-N *GMM*, we learn l_{n+1} for each prefix sequence of N locations, $L_n^N = (l_{n-(N-1)}, \ldots, l_{n-1}, l_n)$, by estimating the various conditional probabilities, $p\left(l_{n+1} = l | L_n^N \right)$. The most commonly used method for estimating these conditional probabilities is to use the maximum likelihood principle, and the conditional probability $p\left(l_i|L_n^N\right)$ is computed by

$$p\left(l_i|L_n^N\right) = \frac{\sharp(L_n^N, l_i)}{\sharp(L_n^N)}, \tag{2}$$

where $\sharp(L_n^N)$ is the number of times that prefix sequence L_n^N occurs in the training set, and $\sharp(L_n^N, l_i)$ is the number of times that location l_i occurs immediately after L_n^N.

Nonetheless, there are some problems with the order-N *GMM*. On one hand, having a fixed N for all cases makes the model too rigid to account for the variability in the characteristics of different sampling locations. On the other hand, for a particular sampling location l_i, the trajectories containing l_i with lengths less than N cannot be used in the model. Therefore, we propose to train a variable-order *GMM*, which can model sequential data of considerable complexity. In contrast to the order-N *GMM*, the variable-order *GMM* learns such conditional distributions with a varying N in response to the available statistics in the trajectories. Thus, a variable-order *GMM* provides the means of capturing different orders of Markov dependencies based on the observed data.

The orders in a variable-order *GMM* are bounded by a pre-determined constant N. To train a variable-order *GMM*, we start with a first-order *GMM*, followed by a second-order *GMM*, etc, until the order-N *GMM* has been obtained. There exist many ways to utilize the variable-order *GMM* for prediction. Here we adopt the principle of longest match to predict next locations. That is, for the given sequence, we first try to make prediction using the order-N *GMM*. If this model does not contain the corresponding state, we then try to predict next locations using the order-$(N-1)$ *GMM*, and so on.

4.2. Personal Markov model

In addition to collective patterns in the trajectories, moving objects often have their own individual movement patterns. For example, the majority of people's movements are routine (e.g., commuting), and most people tend to stick with the routes they are familiar with in the past, even when more reasonable routes exist in certain situations (e.g., traffic jam). In addition, about 73% of trajectories in our dataset contain only one point, but they also can reflect the characteristics of the moving objects' activities. For example, someone who lives in the east part of the city is unlikely to travel to a supermarket 50 km away from his home. Therefore, we propose a Personal Markov Model (*PMM*) for each moving object to predict next locations.

The training of *PMM* consists of two parts: training a variable-order Markov model for every moving object using its own trajectories which contain at least two locations, and a zero-order Markov model for every moving object using the trajectory units.

For training the variable-order Markov model, we construct the prefix sequences for every moving object using its own trajectories, and then we compute the probability distribution of the next sampling locations. Specially, we iteratively train a variable-order Markov model with order ranging from 1 to N using the trajectories of one moving object.

We train a zero-order Markov model using the trajectory units. For a moving object, let $N(l')$ denote the number of times a sampling location l' appears in the training trajectories. Let $\mathcal{L}$ be the set of distinct sampling locations appearing in the training trajectories. Then we have

$$P(l') = \frac{N(l')}{\sum_{l \in \mathcal{L}} N(l)}. \tag{3}$$

The zero-order Markov model can be seamlessly integrated with the variable-order Markov model to obtain the final *PMM*.

4.3. Regional Markov model

GMM uses the trajectories of all moving objects to mine the collective patterns of people. But it makes predictions at too coarse a granularity and does not consider the inherent similarity between different trajectories. Furthermore, similarities often exist between the trajectories in the same region of an urban area. For example, as shown in Fig. 1, compared to T_4, T_1 is more similar to T_3, as they contain the same pattern $\langle u_2, u_3 \rangle$. Based on this observation, we propose to cluster the trajectories and train separate models for each individual cluster. For a given trajectory, we can then find the most relevant cluster and use the corresponding model for prediction. Another benefit of trajectory clustering is that by partitioning the set of trajectories into subsets, we can reduce the number of possible state transitions in the Markov model and make the training more efficient. We call the proposed model the Regional Markov Model (*RMM*).

To train the *RMM*, we first compute the similarity between trajectories and cluster the trajectories accordingly. Then we train a variable-order Markov model for every cluster using the trajectories contained therein. For a given trajectory, we first identify its most relevant cluster and then make prediction of the next location using the corresponding model of the cluster.

For clustering the trajectories, it is necessary to measure the similarity between two trajectories under some metric. Although Euclidean distance and Dynamic Time Warping can be used here, they are all sensitive to noises existing in trajectory data. Therefore, we choose to use Edit Distance on real sequences [32] to measure the distance between two trajectories.

The edit distance between a pair of trajectories T_i and T_j is formalized as follows:

$$ED(T_i, T_j) = \begin{cases} |T_i| & \text{if } |T_j| = 0 \\ |T_j| & \text{if } |T_i| = 0 \\ ED(rest(T_i), rest(T_j)) & \text{if } first(T_i) = first(T_j) \\ \min\{ED(rest(T_i), rest(T_j)) + 1, & \\ \quad ED(rest(T_i), T_j) + 1, & \text{otherwise} \\ \quad ED(T_i, rest(T_j)) + 1\} & \end{cases} \tag{4}$$

where $|T|$ is the length of trajectory T. $ED(T_i, T_j)$ measures the distance between trajectory T_i and T_j, in which $rest(T)$ is the subsequence of T without the first location, and *first*(T) is the first location in trajectory T.

With the distance metric defined, we can perform clustering on the trajectories. As the volume of trajectories is huge and new trajectories are generated continuously, we propose a *Trajectory Clustering* method similar to Trajectory Micro-Clustering used in [33].

Algorithm 1. Trajectory clustering.

```
Input: distance threshold $\delta$, trajectory set $\mathcal{T}$;
Output: the trajectory clusters;
1:  sample some trajectories and initialize clusters
    $C = \{C_1, C_2, \ldots, C_n\}$ through hierarchical clustering;
2:  for every trajectory $T$ in $\mathcal{T}$ do
3:    compute the pairwise edit distance with the representative
      trajectory $T_i^*$ of every cluster $C_i$ according to Eq. (4);
4:    choose the cluster $C_i$ with the minimum edit distance
      $ED(T, T_i^*)$;
5:    if $ED(T, T_i^*) < \delta$ then
6:      add $T$ into $C_i$ and update $C_i$ accordingly;
7:    else
8:      create a new cluster for $T$;
9:    end if
10: end for
11: return the clusters;
```

Algorithm 1 shows the general work flow of trajectory clustering. We first sample some trajectories from the trajectory set and get initial clusters through hierarchical clustering. Each cluster holds and maintains a set of trajectories, and has a representative trajectory, which is the one that has the minimum sum of distances with the other trajectories in the cluster. Next, for each newly arrived trajectory T, we compute its closest cluster C_i. If the distance between T and C_i is less than a distance threshold (δ), T is added into C_i, and the cluster is updated accordingly. Otherwise, a new cluster will be created for T. Finally, the clusters with similar trajectories are returned.

The result of the clustering process is a set of clusters, each containing a set of similar trajectories. We then train a variable-order Markov model with order ranging from 1 to N for every cluster using only the trajectories contained therein, where N is user-specified. When making predictions, we first identify the cluster that is most relevant to the trajectory under consideration, T_{test}, using the following formula:

$$C = \arg\min \, \forall_{i=1,\ldots,N_c} ED\left(T_i^*, T_{test}\right), \tag{5}$$

where N_c is the number of clusters, and T_i^* is the representative trajectory of cluster C_i.

4.4. Integration of models

It has been shown that the combination of multiple predictors can lead to significant performance improvement over the individual algorithms [34]. There are many methods to combine the results from a set of predictors. For our problem, we choose to use linear regression to integrate the models we have proposed.

For the given i-th trajectory sequence, every model can get a vector of probabilities, $\mathbf{p}_i^w = (p_1^i, p_2^i, \ldots, p_m^i)'$ (w represents the wth model), where m is the number of the sampling locations, and p_j^i is the probability of location j being the next sampling location. We also have a vector of indicators $\mathbf{y}_i = (y_1^i, y_2^i, \ldots, y_m^i)'$ for the i-th trajectory sequence, where $y_j^i = 1$ if the actual next location is j and 0 otherwise. We can predict $\mathbf{y}_i$ through a linear combination of the vectors generated by different models:

$$\hat{\mathbf{y}}_i = \beta_0 \mathbf{1} + \sum_{w=1}^{r} \beta_w \mathbf{p}_i^w, \tag{6}$$

where $\mathbf{1}$ is a unit vector and r is the number of models, and β_0, β_w are the coefficients to be estimated.

Given a set of n training trajectories, we can compute the optimal values of β_i through standard linear regression that minimizes $\sum_{i=1}^{n} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|$, where $\|\cdot\|$ is the Euclidean norm. The β_i values thus obtained can then be used for prediction. For a particular trajectory, we can predict the top k next sampling locations by identifying the k largest elements in the estimator $\hat{\mathbf{y}}$.

5. Time factor

The movement of human beings demonstrates a great degree of temporal regularity [35,36]. In this section, we will first discuss how the movement patterns are affected by time, and then show how to improve the predictor proposed in the preceding section by taking the time factor into consideration. Specifically, the choice of the next location is affected by both the sequence of sampling locations that the moving object has just passed, and the time when this movement to next location is made. The sequence of past locations is considered in the Markov model; now we focus on the effect of the time factor.

5.1. Observations and discussions

We illustrate how time could affect people's movement patterns through two examples.

Example 1. Bob is going to leave his house. If it is 8 a.m. on a weekday, he is most likely to go to work. But if it is 11:30 a.m., he is more likely to go to a restaurant, and he may go shopping if it is 3 p.m on weekends.

This example illustrates that people tend to have different movement patterns due to work or life habits.

Next, we use the real dataset consisting of the vehicle passage records to analyze the movement patterns at different time periods.

Example 2. There are seven candidate next locations for a particular location, and the probability distributions of the next sampling locations over seven different time periods are shown in Fig. 2. In this case, the distributions over those locations do differ from one period to another. For instance, vehicles are most likely to arrive at the fifth sampling location during the period from 9:00 to 10:00, whereas the most probable next location is the second for the period from 14:00 to 15:00.

The above examples show that time plays an important role in people's routing choices, and the movement patterns do differ from time to time. As such, the prediction

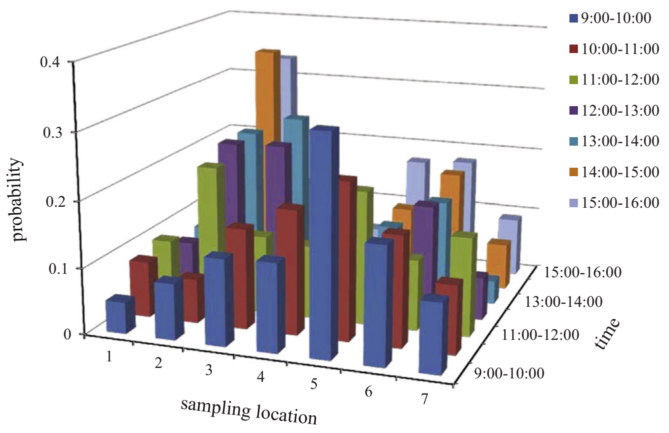

Fig. 2. An example of time affecting people's movement patterns.

model should be made time-aware, and one way to do this is to train different models for different time periods. However, it is non-trivial to determine the suitable time periods. One could attempt to divide the timeline into very fine-grained periods to account for all the changes in movement patterns, but if the time period is too small, there will be only very few trajectories or none in each time period, resulting in severe under-fitting of the models. On the other hand, we cannot always use only one period either, because in general that will be too coarse-grained to capture any changes in movement patterns.

In what follows, we will explore a few methods to tackle this challenge. Here, we choose *day* as the whole time span, i.e., we study how to find movement patterns within a day. However, any other units of time, such as *hour*, *week* or *month*, could also be used depending on the scenario.

5.2. Time binning

A straight-forward approach is to partition the time span into a given number (M) of equi-sized time bins, and all trajectories are mapped to those bins according to their time stamps. A trajectory spanning over more than one bin is split into smaller sub-trajectories such that the trajectory units in each sub-trajectory all fall in the same bin. We then train M variable order Markov models, each for a different time bin, using the trajectories falling in each bin. Prediction is done by choosing the right model based on the time-stamp. We call this approach *Time Binning* (*TB*).

However, this approach has some serious limitations. In particular, the boundaries of the time bins are all fixed once M is given, making it too rigid to partition the time span at the "true" boundaries, i.e., the boundaries where the movement patterns start to change. Moreover, the sizes of all time bins are equal, rendering it difficult to find the correct bin sizes that fit all movement patterns in the time span, as some patterns manifest themselves over longer periods whereas others shorter.

One possible improvement to *TB* is to start with a small bin size, and gradually merge the adjoining time bins if the patterns for those bins are considered similar by some metric. The problem with this approach is that the patterns for the adjoining bins may not be similar to each other, whereas those for the bins apart from each other may be similar instead. For example, in Fig. 2, the distribution for the period of "11:00–12:00" is different from the one for "10:00–11:00"; rather, it is similar to the one from "14:00–15:00" (e.g., they both have the maximum probability at the second sampling location). This calls for a more sophisticated method to take care of the similarities between time bins.

5.3. Distribution clustering

We propose a method called *Distribution Clustering* (*DC*) to perform clustering of the time bins based on the similarities of the probability distributions in each bin. The probability distribution refers to the transition probability from one location to another. Here, we use cosine similarity to measure the similarities between the distributions, but the same methodology still applies when other distance metrics such as the Kullback–Leibler divergence [37] are used.

As the time bins of similar patterns may be different for different locations, we propose to cluster the bins for each location. For an object o appearing at a given sampling location l with a time point falling into the ith time bin, let $\mathbf{P}_i$ be an m-dimensional vector that represents the probabilities of o moving from l to another location, where m is the total number of sampling locations. We measure the similarity of two time bins i and j (with respect to o) using the cosine similarity

$$\cos_{ij} = \frac{\mathbf{P}_i \cdot \mathbf{P}_j}{|\mathbf{P}_i| \cdot |\mathbf{P}_j|}. \tag{7}$$

With the similarity metric defined, we can perform clustering for each sampling location l on the time bins. The algorithm is detailed in Algorithm 2. The results will be a set of clusters, each containing a set of time bins, for the sampling location l.

Algorithm 2. Distribution clustering.

```
Input: cluster number $Q$, time bins number $M$ and the probability
    distributions of trajectories in each time bin;
Output: the clusters;
1: random select $Q$ time bins as the initial cluster centers;
2: repeat
```

```
3:     calculate the similarity of the probability distributions of
       trajectories in each time bin and the cluster centers;
4:     assign each time bin to the cluster center with the maximum
       similarity;
5:     recalculate the probability distributions of trajectories in the
       cluster centers;
6: until clusters do not change or the maximum number of
   iterations has been reached
7: return the clusters;
```

Algorithm 2 illustrates the clustering method for a location l. It chooses Q time bins as the initial clusters (Line 1), and a cluster is defined as a collection of time bins. At each iteration, the time bin is assigned to the cluster with the maximum pairwise cosine similarity (Lines 3–6). When they are merged, the new cluster inherits the probability distribution $\mathbf{P}_i$ of the original cluster i and $\mathbf{P}_j$ of the time bin j, and the probability distribution $\mathbf{P}$ of the new cluster is computed by the following updating rule:

$$\mathbf{P} = \frac{\mathbf{P}_i * Count_i + \mathbf{P}_j * Count_j}{Count_i + Count_j}, \tag{8}$$

where $Count_i$ and $Count_j$ represent the total number of trajectories in cluster i and time bin j respectively.

5.4. Hierarchical clustering

Suppose there exist Q underlying periodic patterns, we can use DC to group the patterns into Q clusters. However, the number of underlying periodic patterns is usually unknown. So we propose a hierarchical agglomerative clustering method to group the patterns. At each iteration of the hierarchical clustering, two clusters with the maximum cosine similarity are merged. We iterate the procedure until the termination condition (e.g., similarity threshold) is met, and the final clusters are obtained. We will describe the clustering method in Algorithm 3.

Algorithm 3. Hierarchical clustering.

```
Input: similarity threshold $\xi$, time bins number $M$ and the
    probability distributions of trajectories in each time bin;
Output: the clusters;
1: initialize $M$ time bins as the initial clusters;
2: repeat
3:     compute the pairwise cosine similarities among every cluster;
4:     merge clusters $i$ and $j$ with the maximum pairwise similarity
       to generate a new cluster;
5:     recalculate the probability distributions of trajectories in the
       new cluster;
6: until $\cos_{ij} < \xi$ or clusters do not change
7: return the clusters;
```

5.5. Next location prediction

After obtaining the final clusters, we start to train the new model with the trajectories of every cluster. For a given location l, supposed we have got Q clusters, we need to train Q variable-order Markov models from the first-order to the Nth-order with trajectories in every cluster. For cluster q, the location l_{n+1} that the object will arrive at next is given by

$$\begin{aligned} l_{n+1} &= \underset{l \in \mathcal{L}}{\arg\max} \left\{ p\left(l_{n+1} = l_i | L_n^N, \mathcal{T}_l^q\right) \right\} \\ &= \underset{l \in \mathcal{L}}{\arg\max} \left\{ \frac{\sharp(L_n^N, l_i, \mathcal{T}_l^q)}{\sharp(L_n^N, \mathcal{T}_l^q)} \right\}, \end{aligned} \tag{9}$$

where $\mathcal{T}_l^q$ is the training set of trajectories in cluster q for location l, and $\sharp(L_n^N, \mathcal{T}_l^q)$ is the number of times that prefix sequence L_n^N occurs in $\mathcal{T}_l^q$, and $\sharp(L_n^N, l_i, \mathcal{T}_l^q)$ is the number of times that location l_i occurs immediately after L_n^N in $\mathcal{T}_l^q$.

In the new models, the sequence of just-passed locations and the time factor are both utilized by combining clustering and Markov model.

6. Performance evaluation

We have conducted extensive experiments to evaluate the performance of the proposed models using a real vehicle passage dataset. In this section, we will first describe the dataset and experimental settings, followed by the evaluation metrics to measure the performance. We then show the experimental results.

6.1. Datasets and settings

The dataset used in the experiments consists of real vehicle passage records from the traffic surveillance system in the City of Jinan with a 6-million population. The dataset contains 10,344,058 records during a period of 31 days (from January 1, 2013 to January 31, 2013). Each record contains three attributes, the license plate number of the vehicle, the ID of the location of the surveillance camera, and the time of vehicle passing the location. There are about 300 camera locations on the main roads. The average distance between a neighboring pair of camera locations is approximately 3 km.

We divide the dataset into three subsets. The training set consists of the records for the first 20 days; the developing set for tuning combination parameters consists of the next 7 days' records; the last 4 days' records are used as the testing set. When testing, we merge the training set and the developing set to train the final predictor.

6.2. Pre-processing

We pre-process the dataset to form trajectories, resulting in a total of 6,521,841 trajectories. As shown in Fig. 3(a), the trajectories containing only one point account for about 73% of all trajectories. We take the rest of 1,760,897 trajectories with the length greater than one to calculate the number of candidate next locations for every sampling location. As shown in Fig. 3(b), about 86.3% of the sampling locations have more than 10 candidate next sampling locations, and the average number of candidate next locations is about 43.

6.3. Evaluation metrics

Our evaluation uses the following metrics that are widely employed in multi-label classification studies [38].

Prediction coverage: It is defined as the percentage of trajectories for which the next location can be predicted based on the model. Let $c(l)$ be 1 if it can be predicted and 0 otherwise. Then $coverage_{\mathcal{T}} = \frac{1}{|\mathcal{T}|}\sum_{l \in \mathcal{T}} c(l)$, where $|\mathcal{T}|$ denotes the total number of trajectories in the testing dataset.

Accuracy: It is defined as the frequency of the true next location occurring in the list of predicted next locations. Let $p(l)$ be 1 it does and 0 otherwise. Then $accuracy_{\mathcal{T}} = \frac{1}{|\mathcal{T}|}\sum_{l \in \mathcal{T}} p(l)$.

One-error: It is defined as the frequency of the top-1 predicted next location not being the same as the true next location. Let $e(l)$ be 0 if the top-1 predicted sampling location is the same as the true next location and 1 otherwise. Then $one-error_{\mathcal{T}} = \frac{1}{|\mathcal{T}|}\sum_{l \in \mathcal{T}} e(l)$.

Average precision: Given a list of top-k predicted next locations, the average precision is defined as $ap_{\mathcal{T}} = \frac{1}{|\mathcal{T}|}\sum_{l \in \mathcal{T}} \frac{p(i)}{i}$, where i denotes the position in the predicted list, and $p(i)$ takes the value of 1 if the predicted location at the i-th position in the list is the actual next location.

6.4. Evaluation of the proposed models

We evaluate the performance of *GMM*, *PMM*, *RMM*, and their different integrations. For each experiment, we perform 50 runs and report the average of the results. In the three variable-order Markov model, there are several user-chosen parameters (e.g., the order N of Markov model, the distance threshold δ in *RMM*) that provide the flexibility to fine tune the model for optimal performance. We study how the choice of these parameter values affects the performance of the models.

First, we study the effect of the order of the Markov model by varying N from 1 to 5 (with fixed $\delta=6$ in *RMM*), and predict top-5 and top-10 next locations with *PMM*, *GMM* and *RMM* respectively. As shown in Fig. 4(a), the accuracy has an apparent improvement when the order N increases from 1 to 2, and starts to decline when N increases further, which indicates that next location is mainly affected by the two preceding locations only. The effect of the order N to average precision is shown in Fig. 4(b), all the models achieves its maximum average precision at the point of $N=2$. As we can see from Fig. 4(c), *PMM* and *RMM* obtain the best performance when N is 2, and *GMM* gets the minimum one-error when N is 3, but it is merely a slight improvement compared with the one-error obtained at $N=2$. As can be observed from Fig. 4(d), the coverages of all the models almost have no change as N increases from 1 to 5, and they are approximately equal to 1. In addition, the size of the variable order of Markov model grows with respect to N. If N gets too large, it may incur a high training cost in terms of time and space. Taking these factors into consideration, we set N to 2 in the following experiments.

We then vary the value of the distance threshold δ in *RMM*, with fixed N ($N=2$), to predict top-5, top-10 next locations and study how it affects the performance of *RMM*. A number of interesting observations can be made from Fig. 5. On one hand, the accuracy, average precision and coverage improve as we increase the value δ, though, after a certain point ($\delta=6$), they start to decrease slightly. On the other hand, the one-error of *RMM* decreases gradually in general with some small fluctuations around $\delta=6$, and the best performance is obtained when δ is set to 6. This suggests that δ should be large enough to put a sufficient number of trajectories in each cluster to reveal meaningful patterns, but not too large to put dissimilar trajectories in the same cluster to affect the prediction accuracy.

Using the optimal values of N and δ, we predict the top-k next locations, and evaluate the performance of the proposed models with the mentioned metrics. We first show the accuracies of all the models for different k in Fig. 6, from which we can observe that the accuracies improve as k increases. The accuracy of *GMM* and *RMM* is significantly better than that of *PMM*, and the best result is given by *RMM*. The reason is that sometimes the individual trajectories are so sparse that *PMM* is not able to detect some latent patterns. *GMM* performs better because it is less susceptible to the data sparsity problem, and may discover collective patterns which have more important influence on the driving routes. The best result is given by *RMM*, since *RMM* clusters trajectories to make the similar movement patterns clearer.

We combine the basic models with linear regression and obtain four composite models *PG* (*PMM*+*GMM*), *PR* (*PMM*+*RMM*), *GR* (*GMM*+*RMM*) and *PGR* (*PMM*+*GMM*+*RMM*). Obviously, the composite models enjoy higher accuracy than the basic models. Furthermore, *PR* performs better than *PG* and *GR*, which indicates that the next locations are mainly affected by both the individual patterns and the aggregated patterns of similar trajectories. When we combine *GMM* and *PR*, the accuracy only has a slight improvement, as the collective patterns are similar to the patterns mined by *RMM*. The optimal prediction accuracy is obtained at $k=10$ by *PGR*. Since

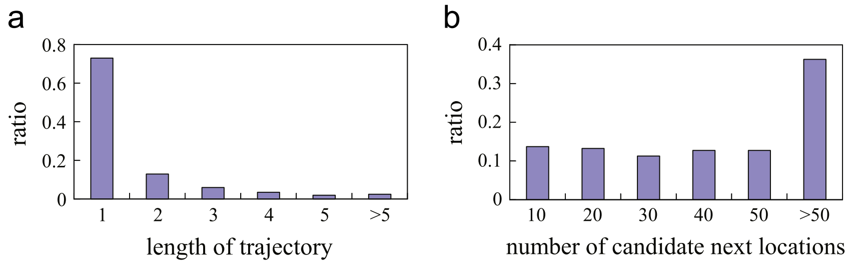

Fig. 3. Characteristics of the Dataset.

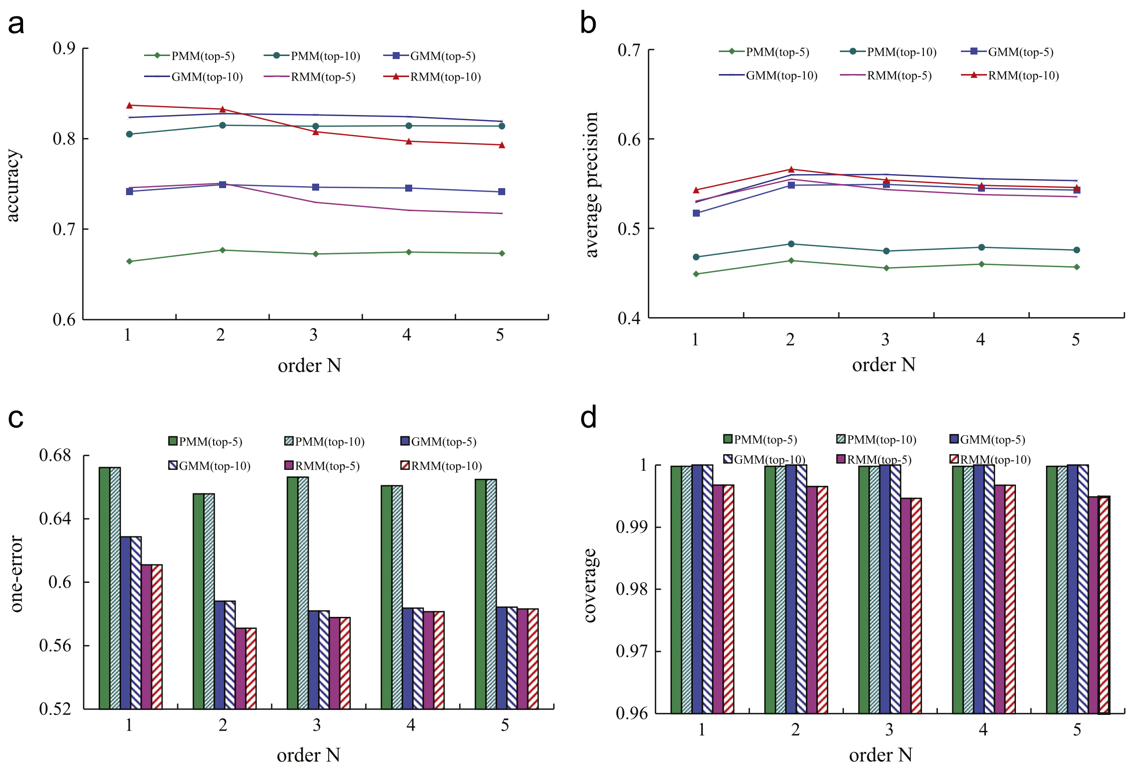

Fig. 4. Effect of the order of Markov model to (a) accuracy, (b) average precision, (c) one-error, (d) coverage.

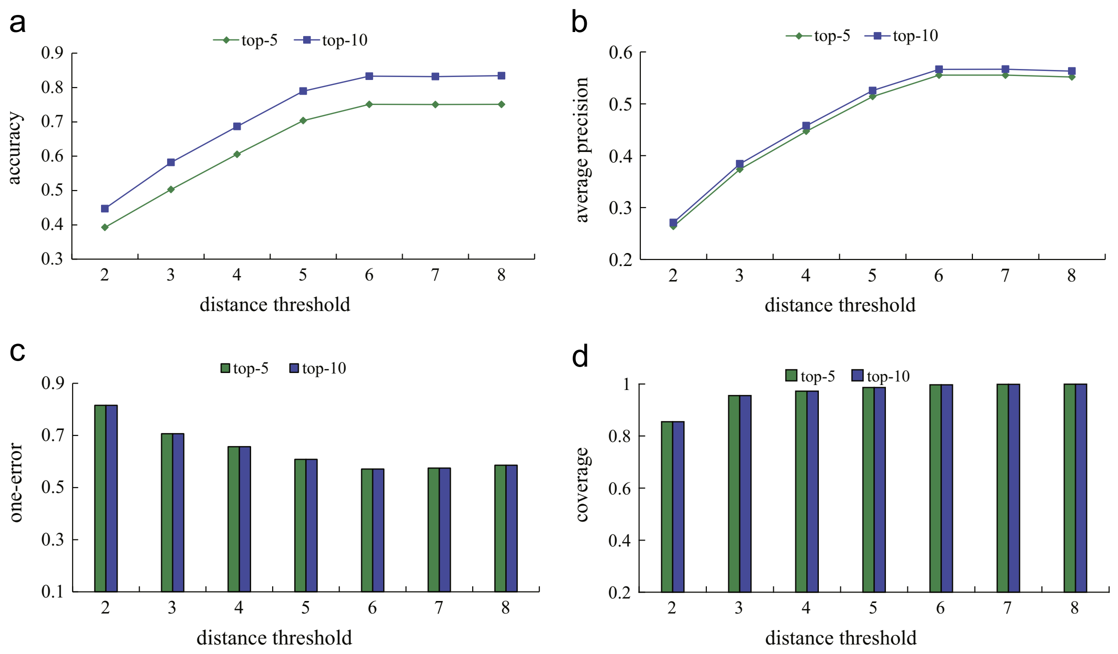

Fig. 5. Effect of the distance threshold of *RMM* to (a) accuracy, (b) average precision, (c) one-error, (d) coverage.

the average number of candidate next locations is 43 (meaning there are 43 possibilities), the accuracy of 90.3% is surprisingly good.

We then set k at 10, and evaluate the coverage, one-error and average precision of the proposed models, as shown in Fig. 7. We observe that the coverages of all models are approximately equal to 1. *GMM* and *RMM* obtain such good performance as they train the model with the global trajectories, while *PMM* obtains high coverage as the training of the zero-order Markov model.

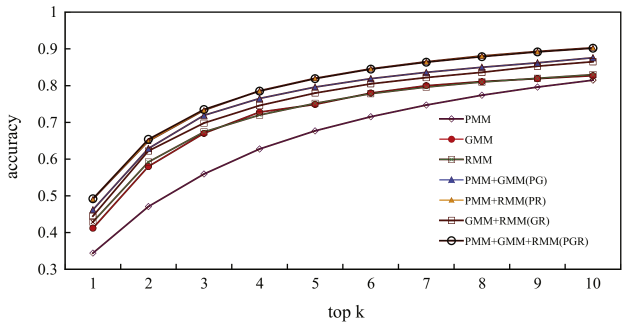

Fig. 6. Accuracy of the proposed models.

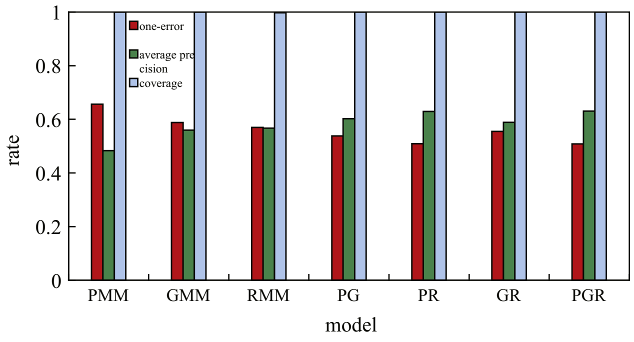

Fig. 7. Coverage, One-error and Average Precision of the proposed models.

In addition, we observe that *PR* and *PGR* are generally superior to the other models, confirming the effectiveness of the integration of models.

6.5. Effect of the time factor

We finally consider the time factor in the optimal model *PGR* and evaluate the performances by predicting top 10 next locations. Fig. 8(a) shows the effect of bin size on *PGR-TB* (which stands for *the integration of PMM, GMM and RMM with Time Binning*). The performance of *PGR-TB* starts to deteriorate when the bin size becomes less than 8, because when the bins get smaller, the trajectories in them become too sparse to generate a meaningful collective pattern. Fig. 8(b) shows the effect of the number of clusters on *PGR-DC* (which stands for *the integration of PMM, GMM and RMM with Distribution Clustering*). When it is set to 1, the model is the same as *PGR*. The one-error rate declines and the average precision improves as the number increases from 1 to 5. When it continues to increase, the result starts to get worse. This is because having too many or too few clusters with either hurts the cohesiveness or the separation of the clusters. Fig. 8(c) shows the effect of different similarity thresholds on *PGR-HC* (which stands for *the integration of PMM, GMM and RMM with Hierarchical Clustering*). The one-error rate declines and average precision improves when the threshold increases to 0.9.

We evaluate the performance of *PGR*, *PGR-TB*, *PGR-DC* and *PGR-HC* with the optimal values of those parameters. As shown in Fig. 8(d), *PGR-TB*, *PGR-DC* and *PGR-HC* perform better than *PGR*, which is because we can get a more refined model by adding the time factor and generate more accurate predictions. *PGR-DC* performs best, validating the effectiveness of the method of distribution clustering. It will be used in the following comparison with alternative methods.

6.6. Comparison with existing methods

We compare the proposed *PGR-DC* with the state-of-the-art approaches *VMM* [13] and *WhereNext* [15]. *VMM* uses individual trajectories to predict the next locations, whereas *WhereNext* uses all available trajectories to discover collective patterns. In this experiment, considering the characteristic of *VMM* and *WhereNext*, we predict top-1 next location. So the one-error and average precision are same as top-1 accuracy, and we will show the performance comparison of *VMM*, *WhereNext* and *PGR-DC* in terms of prediction coverage and top-1 accuracy.

We choose the optimal parameters for *VMM* and *WhereNext* after many experiments. The parameters of *VMM* are set as follows: memory length $N=2$, $\sigma=0.3$, and $N_{\min}=1$. For *WhereNext*, the support for constructing T-pattern tree is set as 20. For the *PGR-DC*, the setting is that the order $N=2$, the distance threshold δ in *RMM* is 6 and the number of clusters in *Distribution Clustering* is set at 5.

As shown in Fig. 9(a), *VMM* has the least prediction coverage, because it only uses individual trajectories and the location sequence of a given trajectory has to match the PST completely in *VMM*, which becomes less frequent when the sampling locations are sparse. In contrast, both *WhereNext* and *PGR-DC* obtain a prediction coverage of 1.0 as the global information is used. Moreover, *VMM* also has

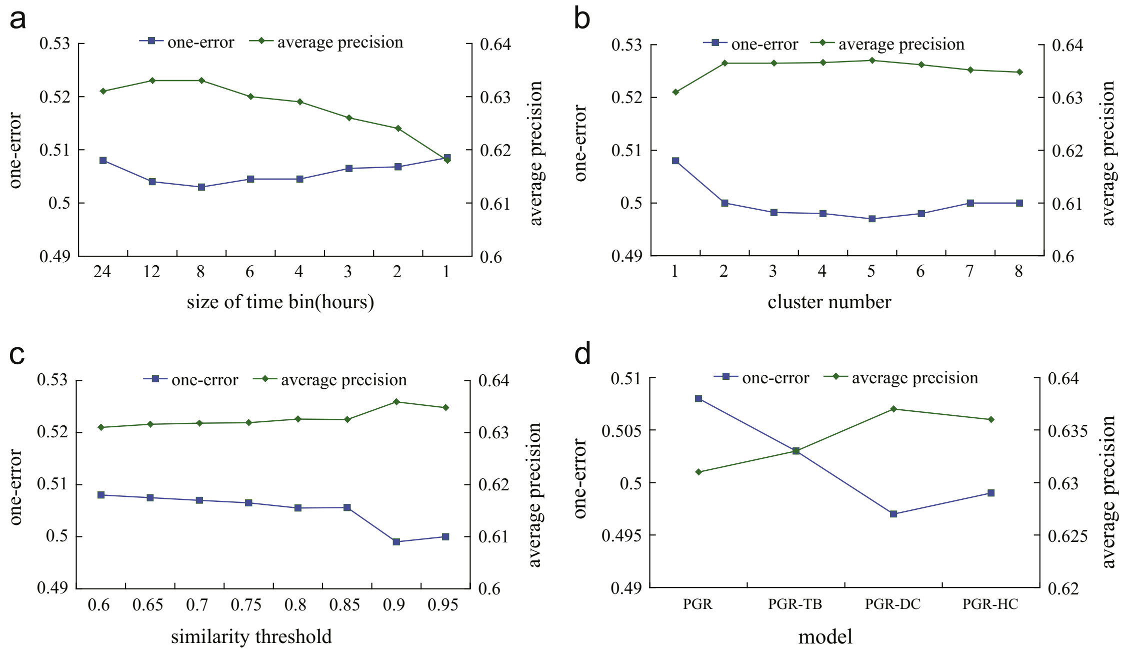

Fig. 8. The effects of parameters on one-error and average precision. (a) Effect of the size of time bin. (b) Effect of the number of clusters. (c) Effect of similarity thresholds. (d) One-error and average precision of different models.

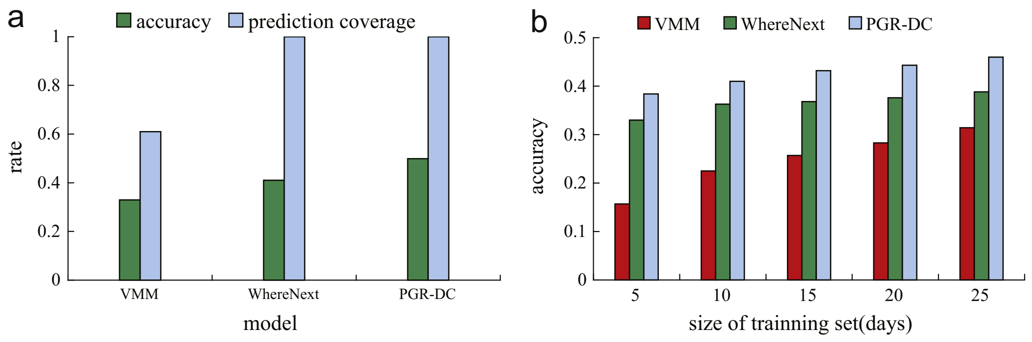

Fig. 9. Performance comparison of *PGR-DC*, *VMM*, and *WhereNext*.

a lower accuracy, as plenty of historical trajectories are pruned away in the process of building PSTs. Without a matching PST, *VMM* cannot make prediction, leading to poor overall accuracy. *WhereNext* makes prediction based on collective patterns, and thus performs better than *VMM*. *PGR-DC* performs the best, which can be attributed to the combination of individual and collective patterns as well as the consideration of time factor. Fig. 9(b) shows that the accuracy of each model improves as the size of training set increases. It is worth mentioning that *PGR-DC* performs better than *VMM* and *WhereNext* in terms of accuracy for any training set size.

7. Conclusion

In this paper, we have proposed three basic models to predict the next sampling location that a moving object will arrive at. *GMM* discovers global behaviors with all available trajectories; *PMM* models the individual patterns of each moving object using its own past trajectories; *RMM* clusters the trajectories to model the regional/local movement patterns. The three models are integrated with linear regression in different ways. The time factor is also added to the models, and we propose three methods to partition the whole time span into periods of finer granularities, including *Time Binning*, *Distribution Clustering* and *Hierarchical Clustering*. New time-aware models are developed accordingly. We have evaluated the proposed models using a real vehicle passage record dataset, and the experiments show that the proposed predictor significantly outperforms the state-of-the-art methods (*VMM* and *WhereNext*).

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China Grant (61272092), the Shandong Provincial Natural Science Foundation Grant (ZR2012FZ004), the Science and Technology Development Program of Shandong Province (2014GGE27178), the 973 Program (2015CB352500), the Taishan Scholars Program, and NSERC Discovery Grants.

References

[1] N.J. Yuan, Y. Zheng, L. Zhang, X. Xie, T-finder: a recommender system for finding passengers and vacant taxis, Trans. Knowl. Data Eng. 25 (10) (2013) 2390–2403.
[2] Y. Zheng, Y. Liu, J. Yuan, X. Xie, Urban computing with taxicabs, in: UbiComp, ACM, Beijing, 2011, pp. 89–98.
[3] T. Liebig, C. Korner, M. May, Scalable sparse Bayesian network learning for spatial applications, in: ICDMW, IEEE, Pisa, 2008, pp. 420–425.
[4] Z. Chen, H.T. Shen, X. Zhou, Discovering popular routes from trajectories, in: ICDE, 2011, pp. 900–911.
[5] L.-Y. Wei, Y. Zheng, W.-C. Peng, Constructing popular routes from uncertain trajectories, in: SIGKDD, 2012, pp. 195–203.
[6] J.-W. Son, A. Kim, S.-B. Park, et al., A location-based news article recommendation with explicit localized semantic analysis, in: SIGIR, ACM, Dublin, 2013, pp. 293–302.
[7] J. Bao, Y. Zheng, M.F. Mokbel, Location-based and preference-aware recommendation using sparse geo-social networking data, in: GIS, ACM, California, 2012, pp. 199–208.
[8] T. Kurashima, T. Iwata, G. Irie, K. Fujimura, Travel route recommendation using geotags in photo sharing sites, in: CIKM, 2010, pp. 579–588.
[9] L. Chen, M. Lv, Q. Ye, G. Chen, J. Woodward, A personal route prediction system based on trajectory data mining, Inf. Sci. 181 (7) (2011) 1264–1284.
[10] H.-P. Hsieh, C.-T. Li, S.-D. Lin, Exploiting large-scale check-in data to recommend time-sensitive routes, in: UrbComp, 2012, pp. 55–62.
[11] T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, K. Fujimura, Geo topic model: joint modeling of user's activity area and interests for location recommendation, in: WSDM, 2013, pp. 375–384.
[12] Z. Yin, L. Cao, J. Han, J. Luo, T. Huang, Diversified trajectory pattern ranking in geo-tagged social media, in: SDM, 2011, pp. 980–991.
[13] G. Xue, Z. Li, H. Zhu, Y. Liu, Traffic-known urban vehicular route prediction based on partial mobility patterns, in: ICPADS, 2009, pp. 369–375.
[14] H. Jeung, Q. Liu, H.T. Shen, X. Zhou, A hybrid prediction model for moving objects, in: ICDE, 2008, pp. 70–79.
[15] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti, Wherenext: a location predictor on trajectory pattern mining, in: SIGKDD, 2009, pp. 637–646.
[16] M. Morzy, Mining frequent trajectories of moving objects for location prediction, in: MLDM, 2007, pp. 667–680.
[17] Z. Li, B. Ding, J. Han, R. Kays, P. Nye, Mining periodic behaviors for moving objects, in: SIGKDD, ACM, Washington, 2010, pp. 1099–1108.
[18] Z. Chen, H.T. Shen, X. Zhou, Y. Zheng, X. Xie, Searching trajectories by locations: an efficiency study, in: SIGMOD, 2010, pp. 255–266.
[19] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, Y. Huang, T-drive: driving directions based on taxi trajectories, in: GIS, 2010, pp. 99–108.
[20] J. Krumm, A Markov model for driver turn prediction, SAE SP 2193 (1), 2008.
[21] J. Froehlich, J. Krumm, Route prediction from trip observations, Soc. Automot. Eng. Spec. Publ. 2193 (2008) 53.
[22] J.A. Alvarez-Garcia, J.A. Ortega, L. Gonzalez-Abril, F. Velasco, Trip destination prediction based on past gps log using a hidden Markov model, Expert Syst. Appl. 37 (12) (2010) 8166–8171.
[23] R. Simmons, B. Browning, Y. Zhang, V. Sadekar, Learning to predict driver route and destination intent, in: ITSC, 2006, pp. 127–132.
[24] E. Horvitz, J. Krumm, Some help on the way: opportunistic routing under uncertainty, in: UbiComp, ACM, Pittsburgh, 2012, pp. 371–380.
[25] J. Krumm, E. Horvitz, Predestination: inferring destinations from partial trajectories, in: UbiComp 2006: Ubiquitous Computing, Springer, California, 2006, pp. 243–260.
[26] B.D. Ziebart, A.L. Maas, A.K. Dey, J.A. Bagnell, Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior, in: UbiComp, ACM, Seoul, 2008, pp. 322–331.
[27] V. Gogate, R. Dechter, B. Bidyuk, C. Rindt, J. Marca, Modeling transportation routines using hybrid dynamic mixed networks, arXiv preprint arXiv:1207.1384.
[28] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma, Mining interesting locations and travel sequences from gps trajectories, in: WWW, 2009, pp. 791–800.
[29] S. Hasan, X. Zhan, S.V. Ukkusuri, Understanding urban human activity and mobility patterns using large-scale location-based data from online social media, in: UrbComp, ACM, New York, 2013, p. 6.
[30] J. Ye, Z. Zhu, H. Cheng, Whats your next move: user activity prediction in location-based social networks, in: SDM, 2013, pp. 171–179.
[31] J.J.-C. Ying, W.-C. Lee, T.-C. Weng, V.S. Tseng, Semantic trajectory mining for location prediction, in: GIS, ACM, Chicago, 2011, pp. 34–43.
[32] L. Chen, M.T. Özsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: SIGMOD, ACM, Maryland, 2005, pp. 491–502.
[33] Z. Li, J.-G. Lee, X. Li, J. Han, Incremental clustering for trajectories, in: DASFAA, Springer, Tsukuba, 2010, pp. 32–46.
[34] M. Jahrer, A. Töscher, R. Legenstein, Combining predictions for accurate recommender systems, in: SIGKDD, 2010, pp. 693–702.
[35] M.C. Gonzalez, C.A. Hidalgo, A.-L. Barabasi, Understanding individual human mobility patterns, Nature 453 (7196) (2008) 779–782.
[36] E. Ben-Elia, Y. Shiftan, Which road do i take? a learning-based model of route-choice behavior with real-time information, Transp. Res. Part A: Policy Pract. 44 (4) (2010) 249–264.
[37] L. Ertoz, M. Steinbach, V. Kumar, A new shared nearest neighbor clustering algorithm and its applications, in: SDM, 2002, pp. 105–115.
[38] M. Ye, D. Shou, W.-C. Lee, P. Yin, K. Janowicz, On the semantic annotation of places in location-based social networks, in: SIGKDD, 2011, pp. 520–528.