Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 824532, 12 pages
http://dx.doi.org/10.1155/2015/824532

Research Article

A Method for Driving Route Predictions Based on Hidden Markov Model

Ning Ye,[1] Zhong-qin Wang,[1] Reza Malekian,[2] Qiaomin Lin,[1] and Ru-chuan Wang[1]

[1]*Institute of Computer Science, Nanjing University of Post and Telecommunications, Nanjing 210003, China*
[2]*Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa*

Correspondence should be addressed to Reza Malekian; reza.malekian@up.ac.za

Received 18 November 2014; Revised 4 January 2015; Accepted 21 January 2015

Academic Editor: Chi-Hua Chen

We present a driving route prediction method that is based on Hidden Markov Model (HMM). This method can accurately predict a vehicle's entire route as early in a trip's lifetime as possible without inputting origins and destinations beforehand. Firstly, we propose the route recommendation system architecture, where route predictions play important role in the system. Secondly, we define a road network model, normalize each of driving routes in the rectangular coordinate system, and build the HMM to make preparation for route predictions using a method of training set extension based on K-means++ and the add-one (Laplace) smoothing technique. Thirdly, we present the route prediction algorithm. Finally, the experimental results of the effectiveness of the route predictions that is based on HMM are shown.

1. Introduction

Currently, many drivers use different kinds of navigation software to acquire better driving routes. The main function of vehicle route recommendation in the software is to find several routes between given origins and destinations by combing some path algorithms with historical traffic data, for example, Google Map and Baidu Map. And then a driver could select one of those recommendation routes according to personal preference, driving distance, and current road congestion information. People usually would like to choose routes with more smooth roads. However, the above methods for driving route recommendation have some problems. Firstly, more people would like to choose routes with many smooth road segments. Thus, the original relatively smooth roads will become congested and the original congested roads will become smooth. Secondly, once a route is selected, the software could not timely inform the driver to adjust the route according to real-time traffic congestion data as the trip progresses. Finally, most of traffic route navigation software programs rely on historical data to predict traffic congestion [1]. While some emergency situations arise, for example, when organizing a large rally in an area, a large number of vehicles will move to this region in a short time, leading to traffic congestion in the area. Obviously, this case may not have happened in previous historical data.

In view of the above problems, a driving route recommendation system is proposed and highlights a method for driving route predictions based on the knowledge of Hidden Markov Model (HMM). The method can predict which road segments are congested or smooth through route predictions. The system will also update traffic information in real time in the near future and inform the driver to adjust the driving route as the trip progresses.

At present, several methods of route prediction have been suggested, but there remain some problems. Karbassi and Barth [2] described a method to predict smart vehicles' routes between given starting and ending drop-off stations based on a car-sharing application. In our work, the destination never needs to be inputted into the system beforehand. Our approach also differentiates from the short-term route prediction in Krumm's work [3]. Our method makes long-term predictions about the entire route. Froehlich and Krumm [4] found that a large portion of a typical driver's trips are repeated from the collected GPS data. So based on this fact, they predicted a driver's entire route by using drivers' trip history. Simmons et al. [5] firstly assumed that drivers have certain routine routes and that, by learning a model based on

previous experience, one can accurately predict what a driver will do in the future. So based on this underlying premise, they presented an approach to predict driver intent using Hidden Markov Models. However, in fact, it is impractical to build a Hidden Markov Model for every driver, and many routes are not fully regular. When a driver takes a new route, the model for this driver could not predict the driver's route and destination intent.

This paper is organized as follows. The next section describes the architecture of our route recommendation system and explains each module in the system. Section 3 introduces how to construct a road network model and Section 4 presents how to define each of the driving routes based on Section 3. The process of building HMM and the method of making route predictions are discussed in Section 5. Then Section 6 shows experimental results. Finally, Section 7 will conclude the paper.

2. The Architecture of Driving Route Recommendation System Based on HMM

The architecture of the driving route recommendation consists of the following phases (see Figure 1).

(i) Driving Route Predictions Based on HMM. It is the core of our recommendation system and is chiefly introduced in this paper. The module could find which routes a driver will be on when making a route prediction. Even though we could not accurately gain the completely correct routes in practice, these possible routes are still very important for preestimating traffic congestion in the future.

(ii) Traffic Congestion Preestimation. It is mainly used to predict the congestion of each road. At the time T_k, the congestion level $RS(T_k, R_i)$ of each road R_i is denoted by the total number of possible driving routes with the road R_i in a time period. The higher the value $RS(T_k, R_i)$ is, the more congested the road R_i is.

(iii) Vehicle Route Recommendation. It collects information about just-driven road segments and traffic congestion situations to introduce better routes for drivers based on existing path algorithms [6–10] (all of these route planning algorithms take traffic congestion situations into account in the process of a vehicle route guidance) without presetting the destination beforehand.

(iv) HMM Correction. It is used to correct the HMM depending on new input driving routes. The given corpus of training samples may not fully include all of possible driving routes. With the increase of inputting driving routes, the amount of training data for training HMM will also grow, which could improve the prediction accuracy.

3. The Definition of Road Network Model

This section will give details on how to build a road network model in the rectangular coordinate system. The connection relationship between roads is followed strictly in the model. And it should reflect the difference between roads as large as possible.

Assume that each road R_i is described as a line segment R_{ix} perpendicular to x-axis: that is, the coordinate of two endpoints of a line segment R_{ix} is separately defined by (X_{i1}, Y_{i1}) and (X_{i1}, Y_{i2}), where $Y_{i1} \neq Y_{i2}$, or a line segment R_{iy} perpendicular to y-axis: that is, the coordinate of two endpoints of a line segment R_{iy} is separately defined by (X_{i1}, Y_{i1}) and (X_{i2}, Y_{i1}), where $X_{i1} \neq X_{i2}$.

In the rectangular coordinate system, the rule for a road network model construction composed of different road segments is represented as follows:

(i) If and only if n ($n \leq 5$) roads $R_{m1}, \ldots, R_{m5}$ intersect at an approximate point, suppose that the road R_{m1} is defined by the line segment R_{m1x} perpendicular to x-axis, so roads R_{m2} and R_{m5} adjacent to the road R_{m1} are represented as line segments R_{m2y} and R_{m5y} intersected with the line segment R_{m1x} and perpendicular to y-axis, and roads R_{m3} and R_{m4} not adjacent to road R_{m1} are separately defined by the line segments R_{m3x} and R_{m4x} intersected with the line segment R_{miy} (R_{m2y} or R_{m5y}) and perpendicular to X. For example, there are five line segments intersected at a point in Figure 2.

(ii) If and only if three different roads R_i, R_j, and R_k intersect at three points (as shown in Figure 3), suppose that the road R_i is defined by the line segment R_{ix} perpendicular to x-axis; then the road R_j is defined by the line segment R_{jy} intersected with the line segment R_{ix} and perpendicular to y-axis, and the road R_k is divided into two segments: one is the line segment R_{kx} intersected with the line segment R_{ix} and perpendicular to x-axis and another is the line segment R_{ky} intersected with the line segment R_{jy} and perpendicular to y-axis.

The length of each line segment is defined as follows: the length of the line segment R_{ix} ($\text{Dist}R_{ix} = |Y_{i2} - Y_{i1}|$) is represented as the amount of line segments perpendicular to y-axis between two endpoints of R_{ix} (including two endpoints), and the length of the line segment R_{iy} ($\text{Dist}R_{iy} = |X_{i2} - X_{i1}|$) is represented as the amount of line segments perpendicular to x-axis between two endpoints of R_{iy} (including two endpoints). But in Figure 3 the length of R_k is different from others. The definitions for the length of R_{kx} and R_{ky} are both limited in the region made up of roads R_i, R_j, and R_k.

Therefore, as shown in Figure 4, our method transforms the map into the road network model in a rectangular coordinate system. Our method only deals with main roads in the map to clearly describe the process of building the model.

4. The Definition of Driving Routes in x-Axis and y-Axis

Suppose that the starting point of the vehicle route is A and the endpoint is B; the route composed of n roads $R_1, R_2, \ldots, R_n$ from A to B is expressed as an ordered

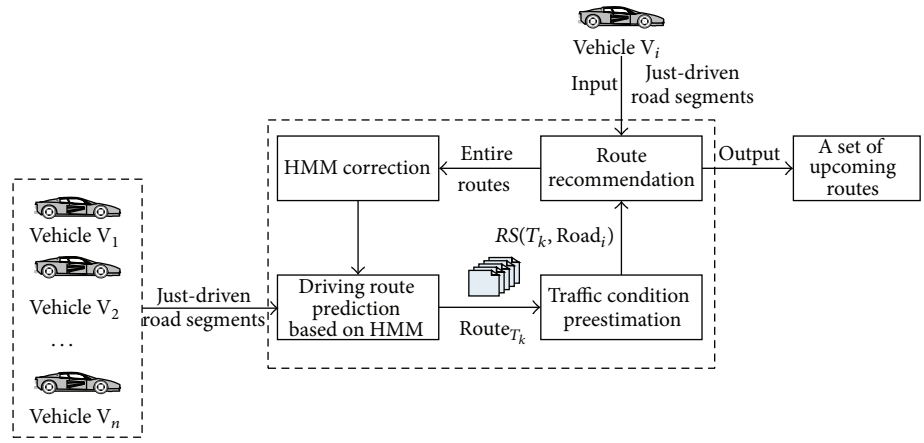


Figure 1: The architecture of route recommendation system.

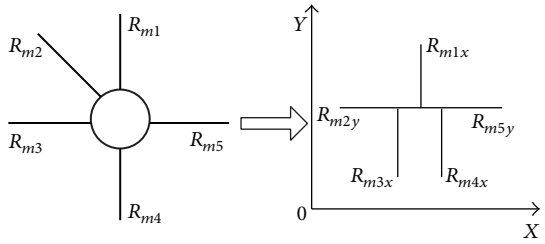


Figure 2: Five roads intersect at a point.

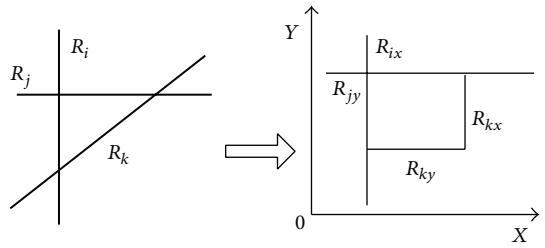


Figure 3: Three different roads intersect at three points.

coordinate points' sequence composed of $n-1$ coordinate points:

$$A \xrightarrow{n} B = R_{1x}\left(R_{1y}\right) \cap R_{2y}\left(R_{2x}\right), \ldots, R_{(n-1)y}\left(R_{(n-1)x}\right) \cap R_{nx}\left(R_{ny}\right), \quad (1)$$

where A is represented as the endpoint of the line segment R_{1x} or R_{1y}, B is represented as the endpoint of the line segment R_{nx} or R_{ny}, and $R_{(i-1)x} \cap R_{iy}$ is represented as the intersection point of the line segments $R_{(i-1)x}$ and R_{iy}.

For example, the line connecting point A (i.e., Huafuyuan) with point B (i.e., Kang'ai Hospital) is a driving route in Figure 5. The vehicle has passed through 5 roads, including Fujian Road, Zhongfu Road, Heilongjiang Road, Jinmao Street, and Xufu Alley. Suppose that A is the starting point and B is the endpoint; then the route can be represented as follows based on Figure 4:

$$\text{Huafuyuan} \xrightarrow{5} \text{Kang'ai Hospital} = (1,3), (1,4), (3,4), (3,1). \quad (2)$$

5. Driving Route Predictions Based on HMM

5.1. A Method of Extending Training Set Based on K-Means++. It is necessary to train the HMM from drivers' past history. In particular, the larger the size of training examples is, the more accurate the HMM for path predictions is. In view of the limitation of given training examples, the training set cannot contain all of routes that drivers will take in the future. So the paper proposes a method of extending training examples based on K-means++ [11]. It could enlarge the training data as much as possible based on given training examples.

After analyzing the given training examples, it is found that starting and endpoints of vehicle routes are distributed in residential, commercial, and work areas. People usually go to work from residential areas in the morning and then go back from work areas or they will first go to commercial areas and then go home. Therefore, it is believed that vehicle routes are generally regular in some extent so that a path can be regarded as two return paths. In addition, it is also found that when traffic reaches its peak, a driver will generally avoid congested roads and select a route with the shortest time to the destination. In other times, drivers will select the shortest distance to the destination to save costs. For a beginning and end of a path, it is able to generate two kinds of routes according to different times.

Last, it is not sure how many clusters the coordinate point set p should be classified beforehand, so the K-means++ algorithm to automatically classify coordinate points into k clusters is exploited in the paper. Here it should be pointed out that the distance of vehicle routes in the same cluster is rather short so that people would not have to drive from one point to another. It is not necessary to calculate vehicle routes for the above case. This assumption will be verified in the experiment.

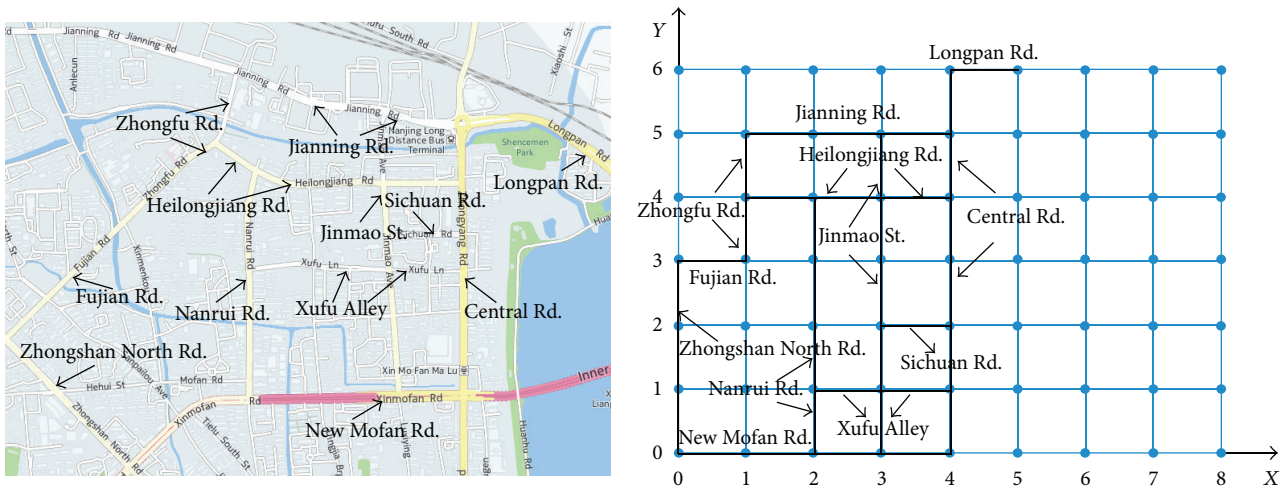


Figure 4: An example of the road network model construction.

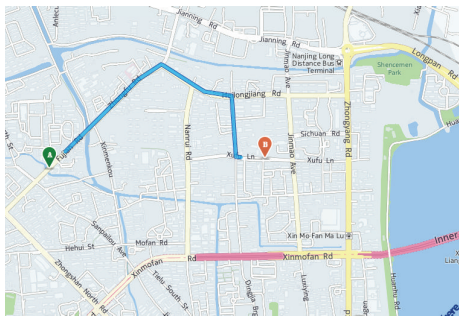


Figure 5: A path between points A and B.

The algorithm of extending training examples based on K-means++ is as follows (see Algorithm 1).

(i) Initialize coordinate point sets p and p' and an extending route set New$D$ (Lines 01-02).

(ii) Traverse a given training set D and read all of vehicle routes' starting points (x_{i1}, y_{i1}) and endpoints (x_{in}, y_{in}), and then insert these coordinate points into the set p. Filter repeated coordinates in the set p, which could get the set p' composed of different starting and endpoints (Lines 03–07).

(iii) Use the K-means++ algorithm to classify p' and then acquire n clusters $C_1, \ldots, C_i, \ldots, C_n$ (Line 08).

(iv) Traverse each cluster C_i and then distinguish whether or not two coordinate points belong to the same cluster C_i. If not, use the function Best_route($c[i][k]$, $c[j][l]$) to calculate routes between two coordinate points (Lines 09–13).

5.2. Parameter Definitions of a HMM for Route Predictions. Since it is necessary to input a driver's just-driven path represented by coordinate points into a HMM and then output future entire paths, coordinate points' sequence corresponding to the just-driven path can be regarded as an observation sequence and the corresponding sequence composed of different route sets can be regarded as a hidden state sequence Q. The next gives details on the process of the HMM construction by following training examples (shown in (3)). Note the number of training examples is much more than following data in practice.

Training Examples. Consider

$$\begin{aligned} t_1 &< (1,3)\,(1,4)\,(3,4)\,(3,1) > \\ t_2 &< (3,1)\,(3,4)\,(1,4)\,(1,3) > \\ t_3 &< (0,3)\,(1,3)\,(1,5)\,(4,5) > \\ t_4 &< (0,3)\,(0,0)\,(0,4)\,(4,1) > \\ t_5 &< (2,0)\,(2,1)\,(3,1)\,(3,2)\,(4,2) > \\ t_1 &< (1,3)\,(1,4)\,(3,4)\,(3,1) > . \end{aligned} \tag{3}$$

In (3), assume that $t_1, t_2, \ldots$ are routes' symbols in order to distinguish different vehicle routes. The observation set V includes the starting symbol (<), the end symbol (>), and different coordinate points. Each observation is defined by p_{ij}, where i is the number of route t_i in the training set and j is the number of coordinate points in each route t_i. For example, the observation set of the above training example is $\{<, >, (1,3), (1,4), (3,4), (3,1), (0,3), (1,5), (4,5), (0,0), (0,4), (4,1), (2,0), (2,1), (3,2), (4,2)\}$. And an observation sequence O is an ordered sequence of symbols and coordinate points from the starting to the end. For example, the observation sequence of the route t_1 is $p_{11} \rightarrow <$, $p_{12} \rightarrow (1,3)$, $p_{13} \rightarrow (1,4)$, $p_{14} \rightarrow (3,4)$, $p_{15} \rightarrow (3,1)$, and $p_{16} \rightarrow >$.

Besides, the definition of hidden states is relatively more complex than observation states. At first, assume that each hidden state is defined by q_{ij}, where i is the number of route t_i in the training set and j is the number of coordinate points in each vehicle route t_i. The hidden state set S includes the symbol • being produced from the observations <, > and different routes' symbol sets (e.g., $\{t_1, t_2, t_3, \ldots\}$) corresponding to different coordinate points. For example, hidden states being produced from the above observations of the route t_1 are separately $q_{11} \rightarrow \bullet$, $q_{12} \rightarrow \{t_1, t_3\}$,

```
Input: A training set D.
Output: The extending training set NewD.
(1)  Coordinate Point Set p, p' = φ;
(2)  Extending route Set NewD = φ;
(3)  foreach (route t_i in D)
(4)     Starting point A = (x_i1, y_i1);
(5)     End point B = (x_in, y_in);
(6)     Insert A and B into the set p;
(7)     p' = Filter(p);
(8)     Cluster Set C = K-means++ (p');
        /* c = {c[1], c[2], ..., c[n]}, which is n clusters altogether. */
(9)  for (int i = 0; i < n; i++)
(10)    for (int j = i + 1; j < n; j++)
(11)      for (int k = 0; k < c[i].length; k++)
            /* c[i].length represents the number of coordinate points in the ith cluster. */
(12)        for (int l = 0; l < c[j].length; l++)
(13)            Insert Best_route(c[i][k], c[j][l]) into NewD;
                /* c[i][k] represents the kth coordinate point in the ith cluster. */
```

Algorithm 1: New_Track (a training set D).

$q_{13} \rightarrow \{t_1\}$, $q_{14} \rightarrow \{t_1\}$, $q_{15} \rightarrow \{t_1, t_5\}$, and $q_{16} \rightarrow \bullet$. A hidden state sequence set is defined by QS, storing hidden state sequences Q being produced from hidden states and each vehicle route is directed. Suppose that $A \xrightarrow{n} B$ represents that a vehicle passes through n road segments from the starting point A to the endpoint B, but $B \xrightarrow{n} A$ represents that a vehicle passes through the same road segments from B to A. Even though each observation state is same in the two opposite routes, ordered coordinate points' sequences are completely opposite. So a method is explored to calculate hidden states corresponding to each coordinate point next.

The algorithm for hidden state determinations is as follows (see Algorithm 2).

(i) Initialize a hidden state sequence set QS (Line 1).

(ii) Obtain a beginning point $A_i(x_{i1}, y_{i1})$ and an endpoint $B_i(x_{in}, y_{in})$ from the vehicle route t_i and a beginning point $A_j = (x_{j1}, y_{j1})$ and an endpoint $B_j = (x_{jn}, y_{jn})$ from the vehicle route t_j; then calculate $\overrightarrow{A_iB_i} = (x_{in} - x_{i1}, y_{in} - y_{i1})$ denoted by $\vec{a}_i$ and $\overrightarrow{A_jB_j} = (x_{jn} - x_{j1}, y_{jn} - y_{j1})$ denoted by $\vec{a}_j$ (Lines 2–9).

(iii) Compute the cosine value of intersection angle between vectors $\vec{a}_i$ and $\vec{a}_j$ (Line 10):

$$\begin{aligned}\cos\left\langle \vec{a}_i, \vec{a}_j \right\rangle &= \frac{\vec{a}_i \cdot \vec{a}_j}{\left|\vec{a}_i\right| \cdot \left|\vec{a}_j\right|} \\ &= \left((x_{in} - x_{i1}) \cdot (y_{in} - y_{i1}) + \left(x_{jn} - x_{j1}\right) \cdot \left(y_{jn} - y_{j1}\right)\right) \\ &\quad \cdot \left(\sqrt{(x_{in} - x_{i1})^2 + (y_{in} - y_{i1})^2} \cdot \sqrt{\left(x_{jn} - x_{j1}\right)^2 + \left(y_{jn} - y_{j1}\right)^2} \right)^{-1}.\end{aligned} \tag{4}$$

(iv) If $0 \le \cos\langle \vec{a}_i, \vec{a}_j \rangle \le 1$, traverse each coordinate point in vehicle routes t_i and t_j, and then judge whether or not a coordinate point o_{k_1} in t_i is also included in t_j. If it is included, insert a symbol t_j into the corresponding location of the sequence Q_i (Lines 10–14). If $-1 < \cos\langle \vec{a}_i, \vec{a}_j \rangle < 0$, driving directions of the two routes are opposite although the routes include the same coordinate point. For example, if a vehicle is driving east in a route t_i, the possibility of passing through south or western roads in a route t_j in our road network model is low. So the kind of hidden states will not be taken into account. And then insert a symbol $\bullet$ and a symbol t_i into Q_i on the basis of the given Q_i (Lines 15–20).

(v) After calculating all of the hidden state sequence, insert each hidden state sequence Q into the sequence set QS (Line 21).

5.3. Parameter Estimation of a HMM for Route Predictions. After determining observation states and corresponding hidden states in the HMM for route predictions, our method uses the total training dataset Total$D$, including the given training set D and the extending training set New$D$, to estimate model parameters. To reduce the negative impact on the HMM, a weighted method is used to improve the process of estimating HMM parameters. In addition, the problem of data sparseness, also known as the zero-frequency problem, arises in the process of building the HMM. So our method adopts the add-one (Laplace) [12] smoothing technique to deal with events that do not occur in the total training set. The process of estimating HMM parameters by a weighted method and add-one (Laplace) smoothing is described as follows.

(i) The following equation is used for the initial probability distribution:

$$\pi_i = \frac{\text{Count}\left(s_{D_i}\right) + \lambda \text{Count}\left(s_{\text{New}D_i}\right)}{\sum_{j=1}^{n}\left[\text{Count}\left(s_{D_j}\right) + \lambda \text{Count}\left(s_{\text{New}D_j}\right)\right]}, \tag{5}$$

Input: A training set D.
Output: A hidden state sequence set QS.
(1) Hidden state sequence set QS = ϕ;
(2) for (int $i = 1$; $i < m$; i++)
/* m is the number of routes in D. */
(3) Starting point $A_i = (x_{i1}, y_{i1})$;
(4) End point $B_i = (x_{in}, y_{in})$;
(5) Vector $\vec{a}_i = (x_{in} - x_{i1}, y_{in} - y_{i1})$;
(6) for (int $j = i + 1$; $j < m$; j++)
(7) Starting point $A_j = (x_{j1}, y_{j1})$;
(8) End point $B_j = (x_{jn}, y_{jn})$;
(9) Vector $\vec{a}_j = (x_{jn} - x_{j1}, y_{jn} - y_{j1})$;
(10) if $(0 \le \cos\langle \vec{a}_i, \vec{a}_j \rangle \le 1)$
(11) foreach (Coordinate point o_{k_1} in t_i)
(12) foreach (Coordinate point o_{k_2} in t_j)
(13) If $(o_{k_1} = o_{k_2})$
(14) Insert a symbol t_j into Q_i corresponding to the coordinate point;
(15) else
(16) foreach (Coordinate point o_j in t_i)
(17) If (o_j is a symbol "<" or ">")
(18) Insert a symbol • into Q_i corresponding to the starting and end point;
(19) else
(20) Insert a symbol t_i into Q_i corresponding to each coordinate point;
(21) Insert each hidden state sequence Q into the sequence set QS

Algorithm 2: Hidden_State_Sequence (a training set D).

where n is the number of hidden states (i.e., the total number of different vehicle routes), Count(s_{D_i}) and Count($s_{\text{New}D_i}$) separately represent the number of times the hidden state s_i appears in the given and extending training sets, and λ represents the weight $(0 < \lambda < 1)$.

(ii) The following equation is used for the hidden state transition matrix:

$$P(s_i \mid s_{i-1}) = \frac{\text{Count}\left(s_{D_{i-1}}, s_{D_i}\right) + \lambda \text{Count}\left(s_{\text{New}D_{i-1}}, s_{\text{New}D_i}\right) + 1}{\text{Count}\left(s_{D_{i-1}}\right) + \lambda \text{Count}\left(s_{\text{New}D_{i-1}}\right) + m}, \tag{6}$$

where Count($s_{D_{i-1}}, s_{D_i}$) and Count($s_{\text{New}D_{i-1}}, s_{\text{New}D_i}$) separately represent the number of times a hidden state s_i followed s_{i-1} in the given and extending training sets and m is the number of times the hidden state s_i occurs in the total training set.

(iii) The following equation is used for the confusion matrix:

$$P\left(v_j \mid s_i\right) = \frac{\text{Count}\left(s_{D_{i-1}}, v_{D_i}\right) + \lambda \text{Count}\left(s_{\text{New}D_{i-1}}, v_{\text{New}D_i}\right) + 1}{\text{Count}\left(s_{D_i}\right) + \lambda \text{Count}\left(s_{\text{New}D_i}\right) + n}, \tag{7}$$

where Count($s_{D_{i-1}}, v_{D_i}$) and Count($s_{\text{New}D_{i-1}}, v_{\text{New}D_i}$) separately represent the number of times the hidden state s_i accompanies the observation state v_j in the given and extending training sets and n is the number of times the observation state v_j occurs in the total training set.

As described above, our method could build the HMM for vehicle route predictions. But drivers would like to choose different vehicle routes from a starting point to an endpoint during different time of each day. For example, people hope to reach the end during the rush hour (7:00~9:00 A.M. and 17:00~19:00 P.M.) as quickly as possible and try their best to avoid congested roads. But at other times people may choose the shortest route to drive. Therefore, training examples can be classified according to the time of day. A group of training examples is from 7:00~9:00 A.M. and 17:00~19:00 P.M., and another is from other times. Section 7 will test the impact on the prediction accuracy with different training examples by building different HMMs at different times.

5.4. Driving Route Predictions. The aim of this section is to introduce how to predict upcoming routes based on just-driven road segments. The solution to this problem is corresponding to a HMM decoding which is to discover the hidden state sequence that was most likely to have produced a given observation sequence. Here, the Viterbi algorithm [13] is used to find the best hidden state sequence composed of different symbols for an observation sequence (a given vehicle route). The process of a vehicle route prediction is shown in Figure 6.

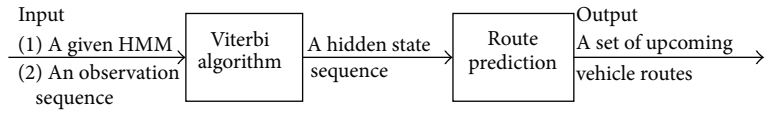


Figure 6: The process of driving route prediction.

Input: An observation sequence O.
Output: A set R of upcoming vehicle routes' symbols.
(1) Ordered Observation Set D_1, $D_2 = \phi$;
(2) Possible Route Set $R = \phi$;
(3) Foreach (Observation p_{ij} in O)
(4) if ($p_{ij} \in V$)
(5) /* V is a set of all of observations in the training set. */
(6) Insert p_{ij} into D_1;
(7) else
(8) Insert p_{ij} into D_2;
(9) int m = length of D_1;
(10) int n = length of D_2;
(11) if ($m = 0$)
(12) $R = \phi$;
(13) else if ($n = 0$)
(14) R = Viterbi_Route ($p_{i1}, p_{i2}, \ldots, p_{ik}$);
(15) else if ($m = 1$ and $D_1(1) = p_{i1}$)
(16) /* $D_1(1)$ represents the first element in the set D_1. */
(17) R = Viterbi_Route (p_{i1});
(18) else if ($D_2(1) = p_{ik}$)
(19) Possible_Routes ($p_{i1}, p_{i2}, \ldots, p_{i(k-1)}$);
(20) else if ($D_2(1) = p_{i1}$)
(21) Possible_Routes ($p_{i2}, \ldots, p_{ik}$);
(22) else
(23) Possible_Routes ($p_{i(j+1)}, \ldots, p_{ik}$);

Algorithm 3: Possible_Routes (an observation sequence O).

Perhaps it will encounter some problems in the process of implementing Viterbi algorithm. The total training set, including the given and extending training examples, is still so limited that it could not fully contain all of possible upcoming vehicle routes. Assuming that the upcoming route does not occur in the total training set, which means (1) part of coordinate points are new ones for training examples and (2) each coordinate point has occurred in the total training set, a group from these coordinate points does not appear in the training examples. For this case (1), the Viterbi algorithm could not be directly used to compute the hidden state sequence. For example, in Figure 5, if a vehicle is on the current road segment represented by $(4, 4)$ and the representation of the corresponding just-driven route is $t_6 < (0, 3)(1, 3)(1, 4)(4, 4)$, the Viterbi algorithm is not adopted to find hidden state sequence for this observation sequence. And for case (2), even though the Viterbi algorithm can be used, each hidden state will not contain this new route's symbol. For example, if a new route is represented by $t_6 < (0, 3)(1, 3)(1, 4)(3, 4)(3, 2)$ and all of these coordinate points have occurred in Figure 5, the symbol t_6 of the upcoming vehicle route will not appear in each hidden state, which means people could not directly understand where the vehicle will drive to. Applied to these problems, an algorithm for vehicle route predictions is proposed as follows (see Algorithm 3).

(i) Suppose that $O = p_{i1}, p_{i2}, \ldots, p_{ik}$ is an observation sequence composed of k coordinate points after the vehicle has passed through k roads; then initialize three sets D_1, D_2, and R, where R represents a set of upcoming vehicle routes' symbols, $D_1 = \{p_{i(x_1)}, p_{i(x_2)}, \ldots, p_{i(x_m)}\}$ ($D_1 \in V$; as described above, V is a set of all of observations in the training set), $D_2 = \{p_{i(y_1)}, p_{i(y_2)}, \ldots, p_{i(y_n)}\}$ ($D_2 \notin V$), and the elements of O are all in the set $D_1 \cup D_2$ (Lines 1-2).

(ii) Traverse the observation sequence O and determine whether or not each coordinate point belongs to the set V. If a coordinate point belongs to V, then insert the point into the set D_1. If not, insert it into D_2 (Lines 3–8).

(iii) Define that m is the number of elements in the set D_1 and n is the number of elements in the set D_2 (Lines 9-10).

(iv) If $m = 0$, the Viterbi algorithm is not used to find the upcoming routes and then $R = \phi$ (Lines 11-12).

```
(1) Hidden state sequence Q = Viterbi(O′);
(2) int m = length of Q;
(3) if (m = 1)
(4)     R = Q_1;
(5) else
(6)     for (int i = 2; i < Num of Q; i++)
(7)         if (R ∩ Q_i ≠ φ)
(8)             R = R ∩ Q_i;
(9)         else
(10)            R = Q_i;
```

Algorithm 4: Viterbi_Route (an observation sequence O').

(v) If $n = 0$, the Viterbi algorithm could be used to predict and then use a function Viterbi_Route to acquire the route set related to the upcoming routes most likely. This set will be helpful for people to drive as much as possible (Lines 13-14).

(vi) If the input observation sequence O has not appeared in the total training set before and part of coordinate points in O have also not appeared in V (i.e., $D_2 \neq \phi$), four cases should be discussed:

(a) Suppose that $D_2 = \{p_{i2}, \ldots, p_{ik}\}$; then possible routes' set could be calculated by the function Viterbi_Route (p_{i1}) (Lines 15–17).

(b) Suppose that $D_2 = \{p_{i(y_1)}, p_{i(y_2)}, \ldots, p_{ik}\}$; then use the function recursion to predict with the observation sequence composed of remaining coordinate points $p_{i1}, p_{i2}, \ldots, p_{i(k-1)}$ (Lines 18-19).

(c) Suppose that $D_2 = \{p_{i1}, p_{i(y_2)}, \ldots, p_{i(y_n)}\}$; then use the function recursion to predict with the observation sequence composed of remaining coordinate points $p_{i2}, p_{i3}, \ldots, p_{ik}$ (Lines 20-21).

(d) In addition to the above cases, suppose that $D_2 = \{p_{i(y_1)}, p_{i(y_2)}, \ldots, p_{i(y_n)}\}$ and $y_1 \neq 1$, $y_n \neq k$, $m \neq 1$; then use the function recursion to predict with the observation sequence composed of remaining coordinate points $p_{i(y_1)}$, $p_{i(y_2)}, \ldots, p_{i(y_n)}$ (Lines 22-23). For example, the input observation sequence is $(0, 3)$ $(1, 3)$ $(1, 4)$ $(4, 4)$ $(4, 5)$, where $(4, 4) \notin V$; then the result of vehicle route prediction is the set of hidden states corresponding to the coordinate point $(4, 5)$.

The function Viterbi_Route is described as follows (see Algorithm 4).

(i) Use Viterbi algorithm to calculate the hidden state sequence Q corresponding to the observation sequence O' (Line 1).

(ii) Define that the number of elements in the hidden state sequence Q is m (Line 2).

(iii) If $m = 1$, a set R of upcoming vehicle routes' symbols is the hidden state set Q_1 (Lines 3-4).

(iv) Calculate the intersection between R and another hidden state set Q_i. If this intersection exists, $R = R \cap Q_i$. If not, $R = Q_i$ (Lines 5–10).

For example, if two hidden states are separately $q_{11} \rightarrow \{t_1, t_3\}$ and $q_{12} \rightarrow \{t_1\}$, then $R = \{t_1, t_3\} \cap \{t_1\} = \{t_1\}$ and the most likely upcoming route is t_1. If two hidden states are separately $q_{11} \rightarrow \{t_3\}$ and $q_{12} \rightarrow \{t_1\}$ and $\{t_3\} \cap \{t_1\} = \phi$, then the most likely upcoming route is t_3.

6. Route Prediction Results

6.1. Experimental Platform. Every vehicle should be equipped with a device for collecting vehicle route data. And data collectors use a mobile phone with software Map Plus. We mainly focus on one of functions, path tracking, to record down the path of driving. It runs in the background, while someone could run other apps or lock the device at the same time. It also can export or send tracked paths as KML files. However, continued use of GPS running in the background can dramatically decrease battery life of mobile phone. So the experiment also needs an external large-capacity battery to support the phone continuously. In addition, researchers install the software Google Earth on the computer to present each of collected vehicle routes.

6.2. Data Collection. A total of 20 volunteers are selected for the purpose of collecting the experimental data. In order to facilitate the communication between volunteers and us, all volunteers are from our university, including 15 teachers and 5 students. A month later our researchers finally acquire a total of 1052 paths, where the number of different routes is 51. The same path is the journey that volunteers start from a point to the end through the same road segments. But in the process of the data collection, there are some problems inevitably.

(i) In tunnels, underground parking, and high-rise dense areas, the phenomenon that part of paths are offset from GPS noise will appear [14].

(ii) Volunteers forget to open the software for recording route data, resulting in collecting route data unsuccessfully.

(iii) Volunteers forget to turn off the software when they drive to the end, resulting in the path to be relatively concentrated in a small area.

Once researchers come across the above problems when checking path data, we will manually correct the GPS data. In summary, the experimental results can overcome the influence of GPS noise and human factor to ensure the accuracy of the collected data.

In the actual process of collecting the GPS data, collective data do not only focus on the longitude and latitude but also combine the GPS data of the starting point, the middle, and the end with road segments, describing the route as a path that is made up of the starting and endpoints and driven streets.

6.3. Experimental Metric. To evaluate the performance of route predictions based on HMM, a metric to explore is the

correct prediction accuracy based on driven process. Suppose that a vehicle has passed through i roads; the possible route set R after predicting based on HMM is $R = \{R_1, R_2, \ldots, R_n\}$. So the definition of the prediction accuracy is as follows:

$$P_i = \frac{\sum_{k=1}^{n} D\left(R_k, CR\right)}{\sum_{t=1}^{n} \text{Dist}\left|R_t\right|} \times 100\%, \tag{8}$$

where CR indicates an entirely upcoming route, $D(R_k, CR)$ represents the number of duplicate road segments between one of possible vehicle routes in the set R—R_k and the entirely upcoming route, and $\text{Dist}|R_t|$ represents the length of the route R_t, that is, the number of road segments.

For example, assume that the total training examples are shown in (3) and t_1 is the upcoming vehicle route, which means CR is t_1 from the starting point $(1, 3)$ to the end $(3, 1)$. When the vehicle has traveled through one road, the observation sequence O is denoted by $O =<, (1, 3)$ and the corresponding hidden state sequence is $Q = \bullet, \{t_1, t_3\}$. So the duplicate between t_1 and t_1, t_3 separately is $D(R_1, R_1) = 6$, $D(R_3, R_1) = 1$. The length of routes R_1 and R_3 is separately $\text{Dist}|R_1| = 6$ and $\text{Dist}|R_3| = 7$. So when the vehicle has passed through the first point, the prediction accuracy is as follows:

$$\begin{aligned} P_1 &= \frac{\text{Repeat}\left(R_1, R_1\right) + \text{Repeat}\left(R_3, R_1\right)}{\text{Dist}\left|R_1\right| + \text{Dist}\left|R_3\right|} \times 100\% \\ &= \frac{6+1}{6+7} \times 100\% = 53.85\%. \end{aligned} \tag{9}$$

6.4. Experimental Results

6.4.1. Training and Test Data. In the experiment, all of collected route examples are from the software Map Plus, where each route is included in a .KML file composed of a series of GPS data. Researchers check these data in a certain time period through Google Earth. According to previous description of the road network model, routes represented by GPS data points could be changed into ones represented by coordinate points.

Besides, some extending training examples are introduced here. These examples are extended from original collected data through a method to enlarge the training set based on K-means++ described before. Firstly, raw training examples composed of coordinate points have been entered. Then all of starting and endpoints can be divided into 5 clusters based on K-means++. It is known that the distance between each coordinate point and the corresponding clustering center is, on average, 0.314 km and the farthest distance between two points in a cluster is, on average, 0.628 km. It can illustrate that the distance between two places in a cluster is relatively short, so most of people would not like to drive. Therefore, this is the reason that extending algorithm was not used to calculate driving route in a cluster.

Figure 7 displays the trip data overlaid on two maps, one of original different routes (a) and the other of original and extending different routes (b). The number of extending training examples is 13605, where the number of routes different from original training examples is 13556.

Finally, the composition of test training examples is illustrated in detail. To test the prediction accuracy of our prediction algorithm, our method should acquire part of real-world vehicle route data. Here the method applies a leave-one-out approach [4, 15], meaning that part of route data are extracted from total training examples as test examples.

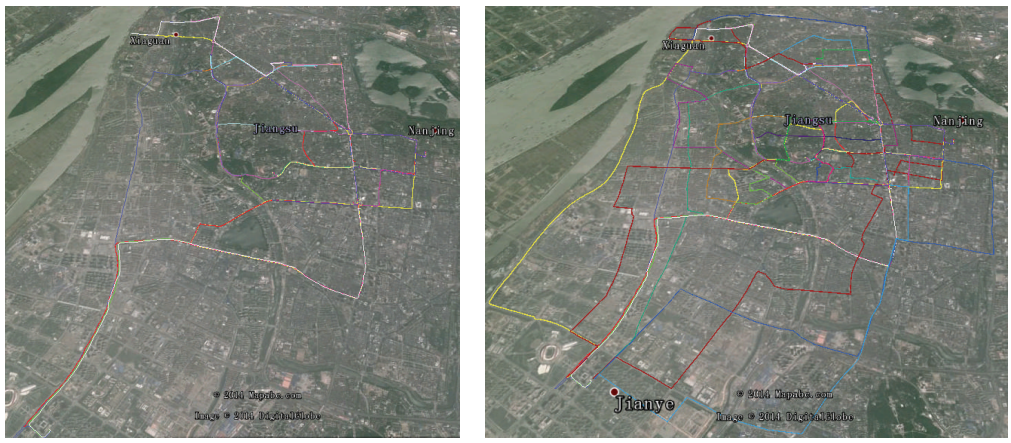
Test Examples (i). It includes part of routes that have not appeared in the training examples. So it can simulate real-world trip data to evaluate the prediction accuracy of our algorithm in actual applications.

Test Examples (ii). All of the route examples have appeared in the training examples. It can evaluate the prediction accuracy compared to test examples (i) in order to illustrate a fact that the number of different routes in the training examples should be as much as possible.

6.4.2. Prediction Accuracy. Figure 8 shows the average correct prediction rate of test examples (i) and test examples (ii) by percent of route completed and by current travel distance with different weight values and also shows the comparison of results between Jon Froehlich's algorithm and our method in these graphs. "Percent of trip completed" is an intuitive evaluation criterion and it is useful in evaluating how well the algorithm performed. However, it is difficult to achieve in practice. A vehicle navigation system can never be sure of how far along a route it is in terms of percentage completed without knowing the exact route of the trip from start-to-end—this is what our prediction method is trying to predict. Instead, a much more practical input parameter is the trip's current distance traveled—that is, how far the vehicle has traveled since the trip began. Furthermore, it also should evaluate the weight value λ to impact HMM for driving route prediction. The algorithm separately set the threshold value λ as 0.2, 0.5, and 0.8.

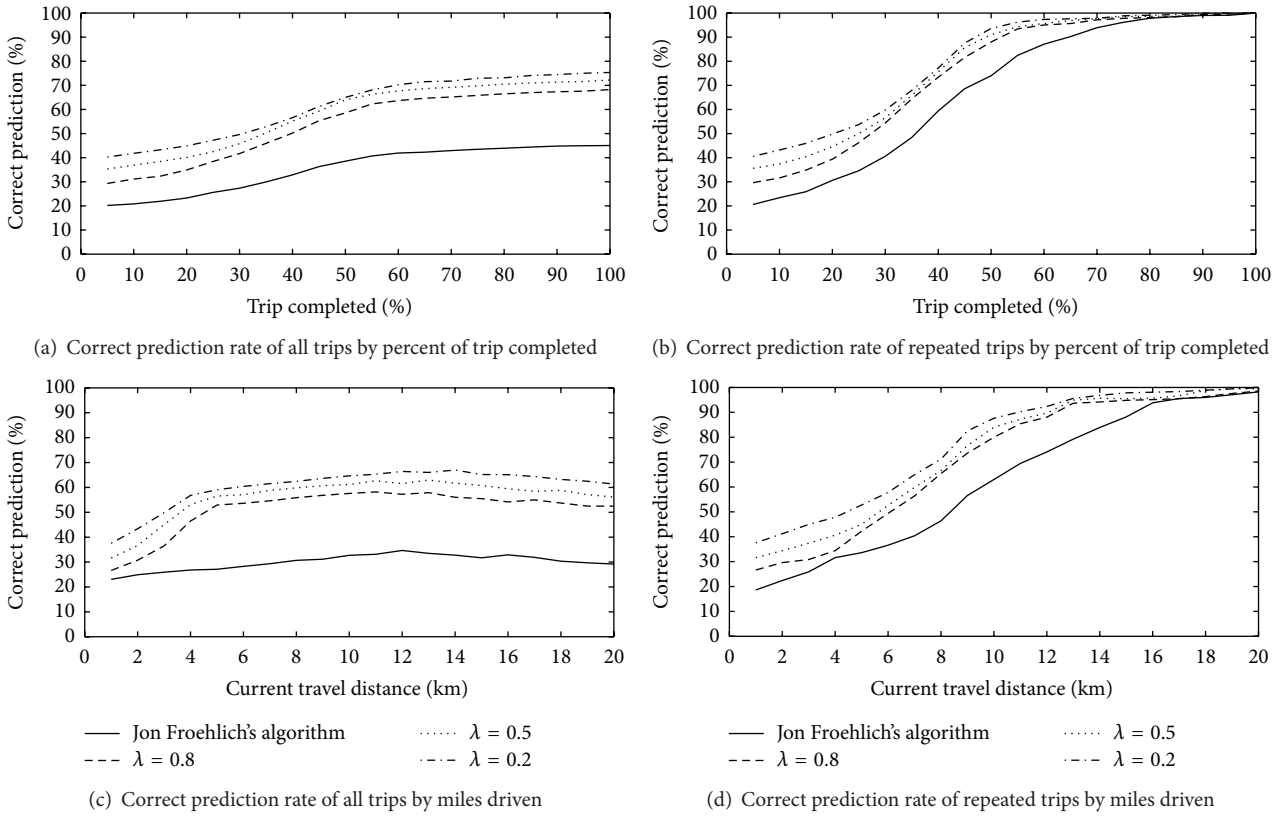
For test examples (i), Figure 8(a) shows that, as expected, after a vehicle has driven the first road segment, little information is known about its path, and the correct prediction rates of both algorithms are much lower. After 35% of the trip has been completed, the correct prediction rate of our algorithm increases to, on average, 49.69% and Jon Froehlich's algorithm only increases to, on average, 29.94%; after 50% completion, the correct prediction rate of our algorithm moves to, on average, 62.52% and Jon Froehlich's algorithm moves to, on average, 38.54%. Figure 8(c) can more accurately show the performance of our proposed algorithm for driving route prediction in a real-world scenario. By the end of the first mile, the correct prediction rate of our algorithm jumps to 31.93% accuracy and by the tenth mile this percentage increases to 61.12%. And the results of Jon Froehlich's algorithm are only between 23.037% and 29.2% for each mile traveled up to 20 miles.

For test examples (ii), Figures 8(b) and 8(d) show that the correct prediction accuracy for both algorithms is, on average, higher than the test dataset (i). In Figure 8(b), the percentage of our algorithm jumps to 90.86% accuracy at the halfway point, but Jon Froehlich's algorithm can increase to this percentage only after 65% of the trip has been completed.



(a) (b)

Figure 7: The trip data overlaid on two maps, one of original data (a) and another of original data and extending data (b).



(a) Correct prediction rate of all trips by percent of trip completed

(b) Correct prediction rate of repeated trips by percent of trip completed

(c) Correct prediction rate of all trips by miles driven

(d) Correct prediction rate of repeated trips by miles driven

Figure 8: The performance of our prediction algorithm and Jon Froehlich's algorithm.

In Figure 8(d), by the end of first mile, the correct prediction accuracy is similar to Figure 8(c), but as the trip progresses, there is a significant jump in prediction accuracy. By the end of 10 miles, the percentage of our algorithm already increases to 83.87%, but at this time Jon Froehlich's algorithm only increases to 63%. As the vehicle has traveled up to 20 miles, the percentage of our algorithm can move to 99.29%.

Figure 8 concludes that the accuracy for driving route predictions increases as the number of observed road segments increases. This means that a longer sequence of road segments will be more helpful for our predictions. Also both of algorithms should take the driving direction into account by the end of first road segment because the vehicle could be heading toward either end of the current road segment and observing only one segment is not indicative of a driver's direction so that the correct prediction rate is nearly zero. Furthermore, the prediction accuracy for repeated trips is already, on average, much higher than for unknown trips.

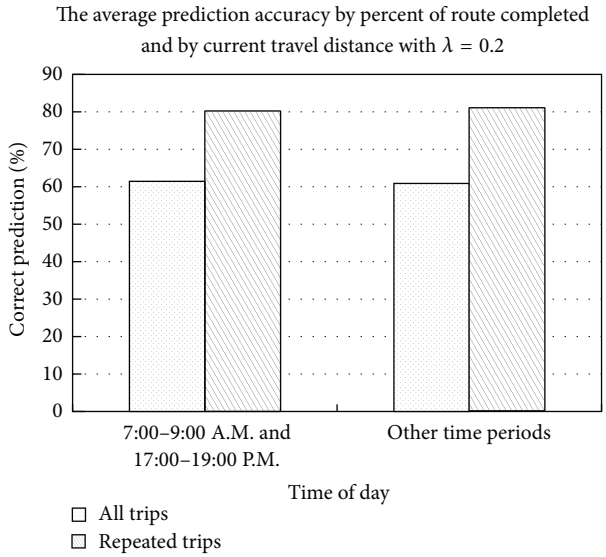


Figure 9: Our algorithm's sensitivity to time of day.

It can demonstrate the necessity of extending the training examples. The probability that new routes occur will be reduced so that the prediction accuracy will be improved as much as possible. At last, the larger the threshold value "λ" is, the lower the correct prediction rate is. In our opinion, driving routes are relatively regular but many route data from extending examples do not follow this rule. Indeed, it will disturb this rule to drop the prediction accuracy. On the other hand, we have to acquire these extending samples, which could improve the prediction accuracy as mentioned before. Therefore, we should keep balance, meaning that extending data not only reduces the impact on a driver's regularity (a regular route is a path that a driver often takes) as much as possible but also keeps it in existence (in the training set) for training and improving the accuracy of HMM. It is similar to core thought of add-one (Laplace) smoothing for the problem of data sparseness. This threshold value is defined as $\lambda = 0.01$ in future applications.

Figure 9 shows the results of prediction accuracy based on different HMMs by the percent of trip completed and by current travel distance depending on the time of day into two categories: (i) 7:00~9:00 A.M. and 17:00~19:00 P.M. and (ii) other time periods. Then, HMMs are trained and tested according to classified test examples. The plot shows that the prediction accuracy is not very sensitive to the time of day, so this is not an important factor to consider when making driving route predictions. Froehlich and Krumm [4] also found a similar lack of sensitivity to both time of day and day of week for increasing prediction accuracy. Above all, it is not necessary to classify training samples to acquire different HMMs for route predictions according to the time of day.

7. Conclusion

This paper firstly presents a driving route recommendation system, where the prediction module is the core of recommendation system, thereby giving details on a method to accurately predict a driver's entire route very early in a trip. Then, a road network model was defined and normalized each of driving routes in the rectangular coordinate system. The method also builds HMMs to make preparation for route prediction using a method of training set extension based on K-means++ and the add-one (Laplace) smoothing technique. Next the paper introduces how to predict upcoming routes in a trip by HMMs and Viterbi algorithm. Finally, experimental results demonstrate the correction of our assumptions as mentioned before and also verify the effectiveness of our algorithm for routes predictions.

As a direction of the future work, the improvement will be from two points: (i) investigate to enhance the Laplace smoothing technique to suit HMM for driving route predictions; (ii) apply the statistics method to make Viterbi algorithm work with unknown coordinate points.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research is support by National Natural Science Foundation of China (nos. 61170065 and 61003039), Peak of Six Major Talent in Jiangsu Province (no. 2010DZXX026), China Postdoctoral Science Foundation (no. 2014M560440), Jiangsu Planned Projects for Postdoctoral Research Funds (no. 1302055C), and Science & Technology Innovation Fund for higher education institutions of Jiangsu Province (no. CXZZ11-0405).

References

[1] A. Hamilton, B. Waterson, T. Cherrett, A. Robinson, and I. Snell, "The evolution of urban traffic control: changing policy and technology," *Transportation Planning and Technology*, vol. 36, no. 1, pp. 24–43, 2013.

[2] A. Karbassi and M. Barth, "Vehicle route prediction and time of arrival estimation techniques for improved transportation system management," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 511–516, IEEE, Columbus, Ohio, USA, 2003.

[3] J. Krumm, "A markov model for driver turn prediction," SAE SP 2193(1), 2008.

[4] J. Froehlich and J. Krumm, "Route prediction from trip observations," SAE SP 2193:53, SAE, 2008.

[5] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, "Learning to predict driver route and destination intent," in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC '06)*, pp. 127–132, IEEE, September 2006.

[6] D. Tian, Y. Yuan, J. Zhou, Y. Wang, G. Lu, and H. Xia, "Real-time vehicle route guidance based on connected vehicles," in *Proceedings of the IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing (GreenCom-iThings-CPSCom '13)*, pp. 1512–1517, Beijing, China, August 2013.

[7] I. Kaparias and M. G. H. Bell, "A reliability-based dynamic rerouting algorithm for in-vehicle navigation," in *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC '10)*, pp. 974–979, IEEE, September 2010.

[8] J.-W. Lee, C.-C. Lo, S.-P. Tang, M.-F. Horng, and Y.-H. Kuo, "A hybrid traffic geographic routing with cooperative traffic information collection scheme in VANET," in *Proceedings of the 13th International Conference on Advanced Communication Technology: Smart Service Innovation through Mobile Interactivity (ICACT '11)*, pp. 1495–1501, IEEE, February 2011.

[9] I. Leontiadis, G. Marfia, D. Mack, G. Pau, C. Mascolo, and M. Gerla, "On the effectiveness of an opportunistic traffic management system for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1537–1548, 2011.

[10] M. H. Kabir, M. N. Alam, and K. K. Sup, "Designing an enhanced route guided navigation for intelligent vehicular system (ITS)," in *Proceedings of the 5th International Conference on Ubiquitous and Future Networks (ICUFN '13)*, pp. 340–344, July 2013.

[11] X. Ma, Y. J. Wu, Y. Wang, F. Chen, and J. Liu, "Mining smart card data for transit riders' travel patterns," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 1–12, 2013.

[12] R. Szalai and G. Orosz, "Decomposing the dynamics of heterogeneous delayed networks with applications to connected vehicle systems," *Physical Review E*, vol. 88, no. 4, Article ID 040902, 2013.

[13] N.-S. Pai, H.-J. Kuang, T.-Y. Chang, Y.-C. Kuo, and C.-Y. Lai, "Implementation of a tour guide robot system using RFID technology and viterbi algorithm-based HMM for speech recognition," *Mathematical Problems in Engineering*, vol. 2014, Article ID 262791, 7 pages, 2014.

[14] B.-F. Wu, Y.-H. Chen, and P.-C. Huang, "A localization-assistance system using GPS and wireless sensor networks for pedestrian navigation," *Journal of Convergence Information Technology*, vol. 7, no. 17, pp. 146–155, 2012.

[15] J. D. Lees-Miller, R. E. Wilson, and S. Box, "Hidden markov models for vehicle tracking with bluetooth," in *Proceedings of the TRB 92nd Annual Meeting Compendium of Papers*, 2013.