International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com

Vol.6., Issue.2, 2018
March-April

RESEARCH ARTICLE

ISSN: 2321-7758

DATA-DRIVEN SIMULTANEOUS TRIP DESTINATION AND JOURNEY TIME PREDICTION

H. X. CAI[1], R. X. ZHONG[2]
[1,2]School of Engineering, Sun Yat-Sen University, Guangzhou, China

H. X. CAI

Renxin Zhong

ABSTRACT

Trajectory data from taxis have become an increasingly important information source for various urban transportation analytics and services, such as human flow dynamics and location-based services. In this paper, we present a novel approach to simultaneously forecast taxi destinations and journey times based on partial trajectories collected from the initial stages of trips. Unlike most existing approaches, which are based on Markov chain models and neural networks that use extensive historical trajectories for training and rule-based methods subject to the knowledge and experience of the researchers, we first utilize the nearest haversine distance of the starting point and the lowest dynamic time warping cost to identify trips with similar patterns. Then, an ensemble machine learning method that combines random forest, support vector regression, and gradient boosted regression is developed to simultaneously predict trip destinations and their journey times by incorporating proper features extracted from real data as input. As validated by empirical results based on a benchmark taxi trajectory dataset from Porto, Portugal, the proposed method can achieve high accuracy in trip destination and journey time prediction.

Keywords: Taxi trajectory data, trip destination prediction, journey time prediction, dynamic time wrapping, ensemble learning

INTRODUCTION

Ten thousand taxis are currently registered in Hong Kong. The spatiotemporal balance between the demand for and supply of taxis is a vital part in providing good access to taxi services in the city due to regulations in taxi registration. The problem of demand–supply matching exists even in other megacities, such as Bangkok, where approximately 100,000 taxis are in service. On the one hand, vacant taxis cruising on roads waste gas and time and contribute to traffic. On the other hand, passengers suffer from waiting longfor taxis in some areas and/or during peak hours. Such an imbalance between demand and supply results in the long-standing challenge in the taxi industry of reducing the number of miles run by taxis without fares, that is, cruising miles, while maintaining a satisfactory level of service on the passenger side. Numerous digital footprints that characterize people's mobility behaviors have become available through the emergence of mobile sensing devices, such as smartphones and GPS navigators. Taxis are now equipped with GPS and communication devices, which periodically upload the status of vehicles (location, speed, presence of passengers, and so on) to the dispatching centers of taxi companies and the transportation department. These digital footprints provide a promising opportunity to understand human mobility patterns and devise a solution to the challenge above.

The current solutions to this problem commonly rely on passengers to provide their location and time of pickup through several communication channels, such as telephone

booking or application-based services. Based on the uploaded vehicle status, dispatching centers schedule taxis (or other on-demand service vehicles) to efficient routes to pick up nearby passengers who actively provide their pickup locations in advance, thus reducing cruising miles. Several on-demand transportation service applications, such as Uber, Lyft, Gett, and Grab, are now offering such location-based services. However, many passengers still hail taxis on streets rather than book services in advance. In addition to achieving better service coverage, taxis should be appropriately pre-positioned at different areas for receiving dispatch orders. Computer science scholars focus on applying computational intelligence and statistics to this challenge by mining taxi trajectory data and categorizing trajectory patterns, such as moving together patterns, trajectory clustering, sequential patterns, and periodic patterns. Several offline prototypes of taxi recommender systems have been developed. To maximize the probability of business success and reduce energy consumption, [1] developed a cost-efficient route recommender system, which can recommend a sequence of pickup locations or potential parking positions to taxi drivers; these recommendations are achieved by learning from the trajectory data transmitted by successful drivers. The T-Finder recommender system developed by [2] provides taxi drivers with a small set of locations and the routes to these locations, thereby allowing the drivers to pick passengers up quickly and maximize the profit of the next trip. Mean while, T-Finder helps passengers by suggesting some locations within reasonable walking distances where they can easily find vacant taxis. Reference [3] proposed T-Share, which extends T-Finder,to large-scale dynamic taxi-sharing systems that accept passengers' real-time ride requests from apps and dispatches taxis to pick passengers up via ridesharing. Knowledge of taxi service mobility patterns (e.g., transporting a passenger from pickup to drop-off locations) and journey time information are essential to these taxi recommender systems. In these recommender systems and other state-of-the-practice on-demand transportation service vehicle dispatching systems, an important but missing component is the simultaneous trip destination and journey time prediction of vehicles for identifying vacant vehicles (in both spatial and temporal domains) in the short-term near future and enabling the system to adapt to dynamic traffic conditions.

Cruising for parking creates a mobile queue of cars, which is a considerable source of congestion. For instance, 30% of moving vehicles are searching for parking with an average searching time of 12min during peak hours in the area around Harvard square in Massachusetts [4]. Although state-of-the-practice parking guidance systems increase the probability of finding vacant parking spaces, drivers may not find vacant parking spots by following the information provided by such systems. Several drivers will possibly go toward the same vacant parking spots, and they may all be occupied by the time the drivers arrive. This situation forces drivers to replan and compete for other spots, thus causing another traffic congestion in areas where parking spaces are monitored. Simultaneous trip destination and journey time prediction by partial trajectory is also important for developing adaptive cruising-for-parking systems in that a system predicts the most possible destinations (proximity to destination) and sends the driver a parking reservation (located in the neighborhood of the destinations to minimize total expected journey time or cost) and route guidance. Agent-based cruising-for-parking simulation platforms have been developed to study the spatiotemporal distribution of parking availability in congested city centers with dynamic characteristics [5]. Reference [6] proposed a resource allocation- and reservation-based smart parking system that assigns and reserves an optimal parking space based on proximity to destination and parking cost. Reference [7] suggested an equilibrium formulation for incorporating parking search in which the search processes employed by drivers are designed to minimize total expected journey time (or cost).

Destination prediction by observing the beginning of a trajectory is also important for many emerging location-based applications, such as sightseeing place recommendations, targeted advertising, automatic destination settings in navigation systems, and early warnings of road congestion. This issue is common in various areas or fields, such as traffic network origin–destination

estimation from limited trajectory trace[8], animal migrations [9], robotic video frames [10], human motion prediction [11], crossroad vehicles [12], and GPS data for studying activity patterns in urban commercial complexes [13]. The rest of this paper is organized as follows. Section 2 states the problem and presents a brief review of related works in this area. Section 3 presents the proposed simultaneous destination and journey time prediction framework. Section 4 discusses the empirical study conducted using a benchmark taxi trajectory dataset. Sections 5 and 6 present the results of the proposed method and the corresponding discussion, respectively. Finally, Section 7 concludes the paper.

Problem statement and related work

A set of vehicle location data, which consists of locations $p(t) \in R^2$ obtained from various observation times t of ongoing taxi tripsis shown in Fig. 1. The blue points are the trip origins (or starting points), the green points are the intermediate observations during the trips, and the red points are the latest observations of each trip. This study mainly aims to develop an efficient framework for simultaneously and accurately forecasting the destination and the associated journey time by observing the initial trajectory of each trip.

The destination prediction problem has been significantly studied. Meanwhile, the literature on simultaneously predicting trip destination and the associated journey time in both computer science and transportation research communities is limited. Destination prediction can be intuitively accessed by comparing known partial trajectories with the current location of trajectories. If an ongoing trip (hereafter called query trajectory) matches part of a popular route derived from historical trajectories, the destination of the popular route is likely to be the destination of the ongoing trip. [14] introduced a neural network (NN) on a feature vector composed of coordinates of the beginnings of trajectories and diverse context information, such as departure time, driver ID, and client information. Their NN algorithm won the 2015 ECML/PKDD discovery challenge (http://www.geolink.pt/ecmlpkdd2015-challenge/) for the destination prediction problem. However, learning performance depends heavily on the professional tuning of the hyper-parameters of the NN. This exposes NN-based approaches to the interpretability problem and renders them unusable in understanding the characteristics of the dataset. Moreover, the training process of the NN is site-sensitive.

Several methods have been proposed based on the Bayesian inference [15, 16].Apart from historical trajectories, [17] incorporated additional information, such as journey time (assuming drivers tend to choose efficient routes in terms of journey time), trip length, accident reports, and driving habits, into the Bayesian inference to compute the probabilities of predicted destinations. These studies aimed to enhance prediction accuracy by using a considerable amount of external information. However, the required external information, such as journey time (to be predicted as well) and driving habits are not always available. The performance deteriorates without sufficient external information. Reference [18] applied a hidden Markov model for destination prediction by observing driver habits through the analysis of GPS data and other information, such as time of day, day of week, and link duration. Reference [19] used GPS data to extract clustered destinations, then applied the hidden Markov model to predict the destination based on the observed partial trajectory. These prediction methods are based on the habits of one or a group of specific individuals based on their historical trip records. Thus, these methods require knowing the identities of drivers. Moreover, the destination can be accurately predicted only when at least one historical record matches the query trajectory [20]. Reference [21] used non-periodic position logs recorded by smartphones to identify user behavior patterns to support destination prediction. Reference [22] proposed the sub-trajectory synthesis algorithm for destination prediction; it addresses the problem of data sparsity while avoiding privacy issues. Reference [23] proposed a nearest-neighbor trajectory method that utilizes distance measures to identify historical trajectories that are similar to the query trajectory and uses the identified trajectory to predict the destination.

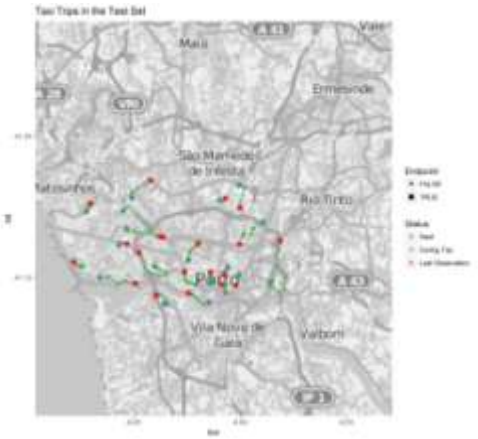

Fig. 1 Trajectory data

Simultaneous trip destination and journey time prediction

A. *Architecture*

Fig. 2 depicts the architecture of the proposed simultaneous trip destination and journey time prediction framework. The basic idea is similarity learning, that is, trajectories with similar route patterns to those of the query trajectory are identified. The similar historical trajectories are then used to predict the possible movements of target vehicles. We start the trip matching by identifying the trips that start closest to the origin of the query trajectory based on the haversine distance. Second, vehicle trajectories are regarded as a time series by applying dynamic time warping (DTW) to search for trips with similar patterns. Third, factors that influence trip destination and journey time are identified. Finally, ensemble machine learning models are utilized to predict trip destinations and journey times.

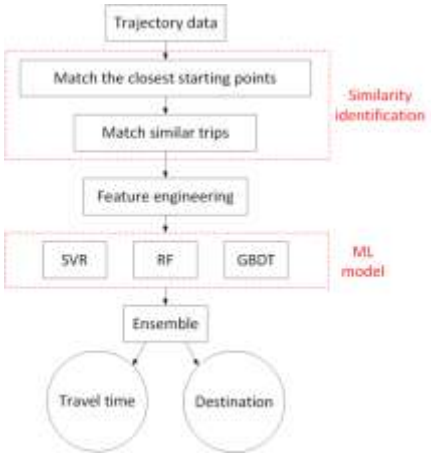

Fig.2 Architecture of simultaneous trip destination and journey time prediction

B. *Closest starting points*

Trips with close starting points are likely to have similar travel patterns than those with geographically distant starting points [24]. To this end, we first define a metric to measure the distance between points on earth. We use the haversine distance to measures distances between two points on earth based on their latitude and longitude.

Definition 3.1 *The haversine distance between two locations* $p_1, p_2 \in R^2$ *on earth is defined as*

$$d(p_1, p_2) = 2 \cdot r \cdot \arctan\left(\sqrt{\frac{a}{1-a}}\right) \quad (1)$$

$$a = \sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right) \quad (2)$$

where (ϕ_1, λ_1) and (ϕ_2, λ_2) are the longitude and latitude of p_1, p_2, respectively, and $r = 6371$ km is the radius of the earth.

C. *Dynamic time wrapping for measuring the distance between two trajectories and similarity learning*

We want to identify several trips at a given trajectory with similar travel patterns to those of the training sample trajectories with the closest starting points to forecast the destination of trips. The divergence between two trips (or trajectories) can be measured by several existing methods, including synchronous euclidean distance [25], closest-pair distance, sum-of-pairs distance [26], and symmetrized segment–pathdistance [27]. The trips are illustrated as time series by adopting the DTW distance to quantify the similarity for two reasons. First, DTW computes the best alignment between two trajectories, which is robust to outliers. Reference [28] Using DTW for driving event recognition shows high accuracy even with a limited training dataset. Second, GPS trajectories are time-dependent sequences with different lengths. DTW has been successfully applied to automatically cope with such time deformations[29]. DTW is a widely applied algorithm that can measure the divergence between two time series with different phases and lengths. This algorithm computes the optimal alignment between two time series under a certain

metric by finding pairs of time indices to align, that is,the optimal warp path. Two given time series can be warped nonlinearly with the obtained warp path in the time domain, and their similarity can be measured easily.

Given two univariate time series $x(i), i = 1, 2, \cdots, m$ and $y(j), j = 1, 2, \cdots, n$,the optimal warp path W can be expressed as

$$W = \begin{pmatrix} w_y(k) \\ w_y(k) \end{pmatrix}, k = 1, 2, \cdots, p \qquad (3)$$

where $w_x(k)$ represents an index from the time series $x(i)$, $w_y(k)$ represents an index from the time series $y(j)$, and p is the length of the warp path W . $\begin{pmatrix} w_x(k) \\ w_y(k) \end{pmatrix}$ indicates that the $w_x(k)$ th element in $x(i)$ corresponds to the $w_y(k)$ th element in $y(j)$.

Several constraints should be satisfied when constructing the warp path W . For example, all indices of both time series should be considered in the warp path W . With this constraint, the warp path W should start at $W(1) = (1,1)$ and end at $W(p) = (m, n)$. Warp path W should be continuous such that adjacent points $W(k)$ and $W(k+1)$ satisfy $w_x(k+1) - w_x(k) \le 1$ and $w_y(k+1) - w_y(k) \le 1$, respectively. Warp path W should be monotonically increasing such that adjacent points $W(k)$ and $W(k+1)$ satisfy $w_x(k+1) - w_x(k) \ge 0$ and $w_y(k+1) - w_y(k) \ge 0$, respectively. Meanwhile, the length of W satisfies $p \in [max(m, n), m+n]$. With the constructed optimal warp path W , the given time series $x(i)$ and $y(j)$ can be extended to two new time series $\bar{x}(k)$ and $\bar{y}(k)$, defined as $\bar{x}(k) = x(w_x(k)), \bar{y}(k) = y(w_y(k)), k = 1, 2, ..., p$, respectively. The warp distance between time series $x(i)$ and $y(j)$ can be represented by a certain distance measure, such as the haversine distance, between these two extended time series $\bar{x}(k)$ and $\bar{y}(k)$ and expressed as $DTW(x, y) = D(\bar{x}, \bar{y}) = \sum_{k=1}^{p} D(x(w_x(k)), y(w_y(k)))$. DTW is optained through the following steps. First, distance matrix C is built; this matrix consists of $M \times N$ elements, andeach element represents the distance between two points in the time series. Second, the accumulated cost distance matrix $D(i, j)$ is constructed, and each element $D(i, j)$ represents the minimum warp distance between the sub-time series x of length i and the sub-time series y of length j . The corresponding path is denoted as W_{ij} . Then, warp path W_{ij} includes $\begin{pmatrix} i \\ j \end{pmatrix}$ and either $\begin{pmatrix} \text{i-1} \\ \text{j} \end{pmatrix}$, $\begin{pmatrix} i \\ j-1 \end{pmatrix}$, or $\begin{pmatrix} i-1 \\ j-1 \end{pmatrix}$, which can construct the relationship between $D(i, j)$ and $D(i-1, j-1)$, $D(i-1, j)$, or $D(i, j-1)$ because $D(i, j)$ represents the minimum cumulative distance from the initial point to $\begin{pmatrix} \text{i} \\ j \end{pmatrix}$ in the cost distance, which is defined as

$$D(i, j) = C(x(i), y(j)) + \min \begin{Bmatrix} D(i-1, j-1), \\ D(i-1, j), D(i, j-1) \end{Bmatrix} \qquad (4)$$

where $D(1,1) = C(x(1), y(1))$. After computing all the elements in the cost distance matrix $D(i, j)$, the optimal warping path can be obtained via a dynamic programming algorithm $DTW(x, y) = \min\{D(m, n)\}$.

Given a test trip trajectory A , supposing M is the number of location samples of A, then the trajectory is denoted as a time series: $A = ([\lambda_{A_1}, \phi_{A_1}], [\lambda_{A_2}, \phi_{A_2}], \ldots, [\lambda_{A_i}, \phi_{A_i}], \ldots, [\lambda_{A_M}, \phi_{A_M}])$. Supposing trip B is one of the trips in the historical data with the closest starting points to trip A , then the length of B is N , which is denoted as $B = ([\lambda_{B_1}, \phi_{B_1}], [\lambda_{B_2}, \phi_{B_2}], \text{K}, [\lambda_{B_i}, \phi_{B_i}], \text{K}, [\lambda_{B_N}, \phi_{B_N}])$. As

mentioned before, λ_{X_i} and ϕ_{X_i} represent the longitude and latitude of the i th location sample in A or B. A pair of trajectories (A, B) is then compared based on the DTW algorithm as previously discussed. First, the cost distance matrix is calculated, where $C(i, j)$ is equal to the haversine distance between $[\lambda_{A_i}, \phi_{A_i}]$ and $[\lambda_{B_j}, \phi_{B_j}]$. Then, the optimal warping path and dynamic time wrapping distance are obtained by a dynamic programming approach.

D. *Feature extraction*

Given n trips with the lowest warping path cost, we can extract several valuable features for destination and journey time predictions. As summarized in Table 1, the spatial trajectory of a taxi can provide some hints on its destination. We extract some features from the trajectory, including the first and the last GPS locations and the distance between adjacent points. The destination is likely in the extension direction of some of the last observed points, so we choose the heading as our feature. The heading between two points is calculated as follows:

$$\theta = atan2(\sin(\phi_2 - \phi_1) * \cos(\lambda_2), \cos(\lambda_1) * \sin(\lambda_2) - \sin(\lambda_1) * \cos(\lambda_2) * \cos(\phi_2 - \phi_1)) \quad (5)$$

where ϕ is the latitude, λ is the longitude, and *atan2* is a common function found in almost all programming languages. θ should be converted to radians before it is used. Meanwhile, the destination of a taxi may be predicted given the taxi ID based on the regularity of pre-hired services. Additional information includes the day of the week, the hour of the day, and exogenous conditions, such as weather or sporting events. Such events can cause a part of the network to behave differently from a typical day. Thus, another feature is whether the data were captured on a holiday. Table 1 shows the selected fetures. We then calculate the themfor each trajectory to generate the feature vectors, which serve as input to the machine learning model that predicts the destination and journey time.

Table 1: List of features

Feature	Explanation	Example
f-lon	First point	–8.600157
f-lat	First point	41.182722
l-lon	Last point	–8.613612
l-lat	Last point	41.183154
distance	Haversine distance of	105.9032
heading	Heading of adjacent points	1.96679
taxi id	Unique identifier for the taxi driver	20000542
wday	Day of week	5
hour	Hour of day	13
holiday	Holiday or not	1

E. *Prediction models*

The taxi destination and the corresponding journey time are predicted using the similarity and the identified significant features by three regression models:support vector regression (SVR), random forest (RF), and gradient boosted regression (GBM).

1) Support vector regression:

A set of training dataset is provided as

$$\{(\mathbf{v}_1, z_1), \cdots, (\mathbf{v}_l, z_l)\}, \quad \mathbf{v}_i \in \mathrm{R}^n, \quad z_i \in \mathrm{R}^1,$$

where v_i represents the feature vector of trajectory, z_i is the corresponding label (journey time or destination), and n is the dimension of the feature vector. The input–output relationship function can be represented by

$$z_i = f(\mathbf{v}_i) = \omega^T \phi(\mathbf{v}_i) + b,$$

where $\phi(\mathbf{v}_i)$ is the non-linear mapping from input feature $\mathbf{v}_i$ to a high-dimensional space, ω is the vector of weightings, and b is a bias term. To train the SVR, we optimize an ε -insensitive loss function with a generalized regression error as follows:

$$\begin{aligned} \min_{\omega,b,\xi,\xi^*} J(\omega,\xi,\xi^*) = \; & \frac{1}{2}\omega^T\omega + K\sum_{i=1}^{l}\xi_i + K\sum_{i=1}^{l}\xi_i^* \\ s.t. \; & \omega^T\phi(\mathbf{v}_i) + b - z_i \le \varepsilon + \xi_i, \\ & z_i - \omega^T\phi(\mathbf{v}_i) - b \le \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \ge 0, i = 1, \cdots, l. \end{aligned} \quad (6)$$

where ξ_i, ξ_i^* are non-negative slack variables, K, ε are the positive regularization parameter and the insensitive loss function parameter, respectively.

Equation (6) is generally solved through its dual problem:

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & \frac{1}{2}(\alpha-\alpha^*)^T Q(\alpha-\alpha^*) \\ & +\varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l} z_i(\alpha_i-\alpha_i^*) \\ s.t. \quad & \mathbf{e}^T(\alpha-\alpha^*)=0, \\ & 0\le\alpha_i,\alpha_i^*\le K, i=1,\mathrm{L}\ ,l \end{aligned} \tag{7}$$

where $Q_{ij}=\mathrm{K}(\mathbf{v}_i,\mathbf{v}_j)\equiv\phi(\mathbf{v}_i)^T\phi(\mathbf{v}_j)$. Then, the estimated values for the SVR model can be rewritten as

$$f(\mathbf{v})=\sum_{i=1}^{l}(-\alpha_i+\alpha_i^*)\mathrm{K}(\mathbf{v}_i,\mathbf{v})+b \tag{8}$$

where $\mathrm{K}(\mathbf{v}_i,\mathbf{v})$ is defined as the kernel function that satisfiesthe Mercer theorem, and α_i,α_i^*, and support vector $\mathbf{v}_i$ are the parameters to be estimated.

Generally, the kernel function can be chosen as linear, polynomial, sigmoid, and radial basis functions (RBF). We adopt the RBF because it has fewer parameters and lower complexitythan the other functions,

$$\mathrm{K}(\mathbf{v}_i,\mathbf{v}_j)=\exp\left(-\gamma\,\mathrm{P}\mathbf{v}_i-\mathbf{v}_j\,\mathrm{P}^2\right),\ \gamma>0.$$

Equation (6) shows that K controls the trade-off between the penalty and the margin. If K is small, the penalty to the samples that exceed the ε range is inadequate, and the regression model is underfitting; otherwise, the model is overfitting, thereby reducing its generalizability. The model becomes more accurate as ε decreases, but at the price of more support vectors and greater complexity.

2) Random forest:

A decision tree builds regression models in the form of a tree structure. To build a decision tree, we should determine which feature used to split the data will bring the maximum information gain. Subsequently, we split the dataset into subsets. Then, we repeat this splitting process on the subsets, until the data in the subset are all of the same class. Decision trees are simple and efficient, but they suffer from highvariancesand tend to overfit.

The bagging algorithm is an ensemble method that constructs several simple estimators on the subsets of the training set, then aggregates their predictions to generate the final result. The bagging method works well for low-bias but high-variance models (e.g., fully developed decision trees) because it provides a way to avoid overfitting.

RF is the combination of the decision tree and bagging methods. The RFmethod utilizes partial sampling data to construct many decision trees while constraining the number of features of each tree to ensure diversity and finally averages the results. RFsignificantly benefits from averaging because its components tend to overfit (decision trees) and are diversified and have low correlation (bagging). This bagging idea can effectively prevent overfitting and become robust with respect to noise [30]. After the training process, we can apply the function to the test set. The prediction for an unseen sample v' in the test set is

$$F(v')=\frac{1}{B}\Sigma_{b=1}^{B} f_b(v') \tag{9}$$

3) Gradient boosted regression:

TheGBMis another ensemble method that uses a decision tree. It is a boosted algorithm because a new model is added to the base model to correct the residual errors of theoriginal model. Boosting aims to build a powerful model by combining several weak models. The formulation of the gradient boosting machine is

$$F(v)=\sum_{m=1}^{M}\gamma_m h_m(v) \tag{10}$$

where $h_m(v)$ are the basis functions, which in our case are decision trees with fixed sizes and are also called weak learners. γ_m is the step length of a gradient descent in the iteration. Given the current model $F_{m-1}(v)$ and label z_i at each iteration, decision tree $h_m(v)$ is trained to minimize loss function L (least squares in our case) and obtain the new model $F_m(v)$:

$$\begin{aligned} & F_m(v)=F_{m-1}(v)+ \\ & \arg\min_{h}\sum_{i=1}^{n} L(z_i,F_{m-1}(v_i)-h(v)) \end{aligned} \tag{11}$$

This minimization problem can be solved with gradient descent, which is the acquisition of the

minimum loss function values along the negative gradient descent direction.

$$F_m(v) = F_{m-1}(v) + \gamma_m \sum_{i=1}^{n} \nabla_F L(z_i, F_{m-1}(v_i)) \quad (12)$$

After training the function $F(v)$ using the training data, the prediction of the unseen test data v' is $F(v')$.

4) Ensemble:

Ensemble is an excellent way to reduce generalization error and improve predictive performance, especially when ensemble model members have low correlations [31]. To improve the stability and accuracy of the result, we usedan ensemble of three base models described earlier. The three commonly used ensemble approaches are ranking averaging, weighted averaging, and majority voting [32,33]. We compared the performances of the three algorithms and finally chose weighted averaging as our ensemble strategy. Fig. 3 shows the ensemble architecture, which is composed of two levels. Level 1 consists of RF, SVR, andGBM, which are merged in level 2. Weight c_i is calculated based on theperformances of the ensemble members and $\sum_{i=1}^{3} c_i = 1$.

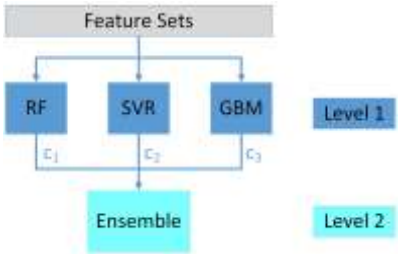

Fig.3Illustration of the ensemble

I. CASE STUDY

*F. **Dataset description***

We use the data provided by Kaggle ECML-PKDD 2015 competition (https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data), which provides an accurate dataset of the trajectories for 442 taxis running in Porto, Portugal. Each data sample represents one completed trip. We only use the non-personalized information to promote the extendability of our method, and the seven features considered are as follows:

1. TRIP_ID: (string) Unique identifier for each trip;
2. CALL_TYPE: (char) Identifies the way the service is demanded and may contain one of three possible values: trip dispatched from central, demanded directly from a taxi driver on a specific stand, and demanded on a random street;
3. ORIGIN_STAND: (integer) Unique identifier for the taxi stand;
4. TAXI_ID: (integer) Unique identifier for the taxi driver;
5. TIMESTAMP: (integer) Unix timestamp (in seconds) that identifies the trip's start;
6. DAYTYPE: (char) Identifies the daytype of the trip's start and is one of three possible values: holiday, the day before a holiday, and normal day;
7. POLYLINE: (string) Contains a list of GPS coordinates mapped as a string and has one pair of coordinates for every 15 seconds of the trip;, the first and last items correspond to the start andthe trip's destination, respectively.

The whole dataset is divided into two parts: training and testing sets. The training set is used to establish a model, and the testing set is used to evaluate the performance of our framework on unseen data. The training set contains a full year of data, with dates ranging from 01/07/2013 to 30/06/2014. The testing set contains 320 taxi trips between 01/07/2014 and 31/12/2014, and partial trajectories are provided. With only 320 trips, the test set is small. Therefore, we randomly select 1500 trips from the training set as our validation set, which tunes the model parameters and determines the stopping point. Furthermore, the validation set can test the robustness of our solutions.

A. Cluster performance

Fig. 4 shows an example of the coordinates of trips in the testing dataset and corresponding trips in the training dataset with low warping path costs. The trips extracted from the training dataset always have the same heading, and their destinations are in a small region.

Fig. 4 Clusteringresults

B. *Feature importance*

The importance of features is calculated based on RF to estimate the influence of these features on journey time. As Fig. 5 shows, the heading of adjacent points has a great influence on journey time prediction, and the last point's location has a greater impact on the result than the first point's location.

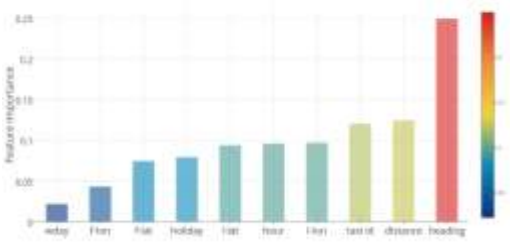

Fig. 5Importance of features

Fig. 6 shows the destination heat map of the 3103 trips of taxi ID 20000610. Three major destinations are dominant, and we can narrow down the possible destinations using the taxi ID.

Fig. 6 Heatmap of destinations of taxi ID = 20000610

Results

A. *Evaluation metric*

The evaluation metric for destination prediction is the mean haversine distance (MHD) of all predicted trips. The haversine distance is introduced in Formulas 1 and 2. The root mean square logarithmic error (RMSLE) evaluates the travel time prediction, which is calculated as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2} \quad (13)$$

where n is the number of hours in the test set, p_i is the predicted trip time, a_i is the actual trip time, and $\log$ is the natural logarithm.

B. *Performance comparison with single models*

Each experiment was conducted 50 times to compare the results of the different models and the different n values (the number of lowest DTW distance trips applied to training). The corresponding average results for the validation set are shown in Table 2.

Table 2: Taxi trip prediction result

	RF	SVR	GBDT	Ensemble
n=25(RMSLE)	0.50710	0.50081	0.50895	0.49605
n=25(Mean Haversine Distance)	2.51615	2.34878	2.31642	2.30393
n=50(RMSLE)	0.49650	0.48305	0.48028	**0.48001**
n=50(Mean Haversine Distance)	2.48480	2.33148	2.27833	**2.27797**
n=75(RMSLE)	0.49650	0.48305	0.48153	0.48028
n=75(Mean Haversine Distance)	2.48480	2.33148	2.27945	2.27833
n=100(RMSLE)	0.49792	0.48312	0.50405	0.48294
n=100(Mean Haversine Distance)	2.47551	2.32685	2.28062	2.27942

C. *Performance comparison over different leader boards*

The testing set was divided into two parts: public and private. The public leader board provides feedback, but the final scores were calculated from the private leader board. The private leader board was calculated with approximately 50% of the test data. Table 3 compares our best ensemble results with the results of the top three in the private leader board. The trip time and destination in our results would have been at the 5th and 22nd positions in the private leader board, respectively, using the ensemble result with the 50 closest trips

and can serve as benchmarks for these two problems.

Table 3: Public and private leaderboards

	PublicRMSLE	PrivateRMSLE	PublicMHD	PrivateMHD
First-placeresults	0.50391	0.52528	2.53217	2.03489
Second-placeresults	0.53252	0.52787	2.36446	2.08772
Third-placeresults	0.49408	0.53097	2.44518	2.11751
Ourresults	0.53166	0.53912	2.33258	2.24419

Discussion

From Table 2, we can draw the following conclusions. First, the results varied for different n; a suitable n can guarantee sufficient data samples without introducing excessive outlines. In our case, the best results were achieved when n=50. Second, the proposed ensemble model was compared with single models. The ensemble model outperformed the single models in both tasks due to the significant diversity among the three models.

Table 3 shows that the champion teams in taxi destination prediction do not always score highly in the public leader board in both tasks, which means that most of other commits tend to over fitting. Our results are consistent between both leader boards, which illustrates the robustness of our method.

Conclusion

We propose a data-driven predictive framework, which consists of DTW and ensemble learning, to simultaneously predict taxi trip destination and journey time. The DTW algorithm is used to extract similar paths and ensemble learning is used to predict the destination and journey time. The experimental results using a taxi trajectory dataset from Porto, Portugal show that our models predict both variables well. Unlike NN, our method does not need map matching and artificial experience. Furthermore, our model is faster and more interpretable than NN. Finally, our algorithm can be easily applied to other datasets because it does not utilize personal information (thereby upholding privacy protection) and the required data are readily available and easy to collect.

References

[1] Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser, M., and Pazzani, M., 2010. An energy-efficient mobile recommender system. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, Dc, Usa, July, pages 899–908.

[2] Yuan, N. J., Zheng, Y., Zhang, L., and Xie, X., 2013. T-finder: A recommender system for finding passengers and vacant taxis.Knowledge and Data Engineering IEEE Transactions on, 25(10):2390–2403.

[3] Ma, S., Zheng, Y., and Wolfson, O., 2013. T-share: A large-scale dynamic taxi ridesharing service. In IEEE International Conference on Data Engineering, pages 410–421.

[4] Shoup, D. C., 2006. Cruising for parking. Transport Policy, 13(6):479–486.

[5] Levy, N., Martens, K., and Benenson, I., 2012. Exploring cruising using agent-based and analytical models of parking. Transport- metrica, 9(9):773–797.

[6] Geng, Y. and Cassandras, C. G., 2011. New smart parking system based on resource allocation and reservations. Intelligent Transportation Systems IEEE Transactions on, 14(3):1129–1139.

[7] Boyles, S. D., Tang, S., and Unnikrishnan, A., 2015. Parking search equilibrium on a network. Transportation Research Part B:Methodological, 81:390–409.

[8] Calabrese, F., Lorenzo, G. D., Liu, L., and Ratti, C., 2011. Estimating origin-destination flows using mobile phone location data. IEEE Pervasive Computing, 10(4):36–44.

[9] Gaffney, S. and Smyth, P., 1999. Trajectory clustering with mixtures of regression models. In ACM SIGKDD InternationalConference on Knowledge Discovery and Data Mining, pages 63–72.

[10] Kruse, E., Gutsche, R., and Wahl, F. M., 1997. Acquisition of statistical motion patterns in dynamic environments and theirapplication to mobile robot motion planning.

[11] Vasquez, D. and Fraichard, T., 2004. Motion prediction for moving objects: a statistical approach. In IEEE International Conference on Robotics and Automation, pages 3931–3936 Vol.4.

[12] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., and Maybank, S., 2006. A system for learning statistical motion patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(9):1450–64.

[13] Kim, B., Ha, J. Y., Lee, S. J., Kang, S., Lee, Y., Rhee, Y., Nachman, L., and Song, J., 2011. Adnext: a visit-pattern-aware mobile advertising system for urban commercial complexes. In The Workshop on Mobile Computing Systems and Applications, pages 7–12.

[14] de Bre´bisson, A., Simon, E´ ., Auvolat, A., Vincent, P., and Bengio, Y., 2015. Artificial neural networks applied to taxi destination prediction. arXiv preprint arXiv:1508.00021.

[15] Krumm, J. and Horvitz, E., 2006. Predestination: inferring destinations from partial trajectories. In UBICOMP 2006: Ubiquitous Computing, International Conference, UBICOMP 2006, Orange County, Ca, Usa, September, pages 243–260.

[16] Krumm, J. and Horvitz, E., 2007. Predestination: Where do you want to go today? Computer, 40(4):105–107.

[17] Krumm, J., 2006. Real time destination prediction based on efficient routes. Technical report, SAE Technical Paper.

[18] Simmons, R., Browning, B., Zhang, Y., and Sadekar, V., 2006. Learning to predict driver route and destination intent. In 2006 IEEE Intelligent Transportation Systems Conference, pages 127–132. IEEE.

[19] Alvarez-Garcia, J. A., Ortega, J. A., Gonzalez-Abril, L., and Velasco, F., 2010. Trip destination prediction based on past gps logusing a hidden markov model. Expert Systems with Applications, 37(12):8166–8171.

[20] Xue, Y., 2015. Recommendation Systems for Travel Destination and Departure Time. PhD thesis, The University of Melbourne.

[21] Nakahara, F. and Murakami, T., 2012. A destination prediction method based on behavioral pattern analysis of nonperiodic position logs. In 6th Inter. Conf. on Mobile Computing and Ubiquitous Networking.

[22] Xue, A. Y., Zhang, R., Zheng, Y., Xie, X., Huang, J., and Xu, Z., 2013. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pages 254–265. IEEE.

[23] Tiesyte, D. and Jensen, C. S., 2008. Similarity-based prediction of travel times for vehicles traveling on known routes. In Pro- ceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems, page 14. ACM.

[24] Zhang, J., Zheng, Y., Qi, D., Li, R., and Yi, X., 2016. Dnn-based prediction model for spatio-temporal data. In ACM Sigspatial International Conference on Advances in Geographic Information Systems.

[25] Potamias, M., Patroumpas, K., and Sellis, T., 2006. Sampling trajectory streams with spatiotemporal criteria. In 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), pages 275–284. IEEE.

[26] Zheng, Y. and Zhou, X., 2011. Computing with spatial trajectories. Springer Science & Business Media.

[27] Besse, P. C., Guillouet, B., Loubes, J.-M., and Royer, F., 2016. Destination prediction by trajectory distribution based model. arXiv preprint arXiv:1605.03027.

[28] Johnson, D. A. and Trivedi, M. M., 2011. Driving style recognition using a smartphone as a sensor platform. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 1609–1615. IEEE.

[29] Mu¨ller, M., 2007. Dynamic time warping. Information retrieval for music and motion, pages 69–84.

[30] Breiman, L., 2001. Random forests. Machine Learning, 45(1):5–32.

[31] Oza, N. C., 2000. Online ensemble learning. In Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, Usa, page 1109.

[32] Breiman, L. and Breiman, L., 1996. Bagging predictors. In Machine Learning, pages 123–140.

[33] Yu, L., Wang, S., and Lai, K. K., 2008. Credit risk assessment with a multistage neural network ensemble learning approach. Expert Systems with Applications, 34(2):1434–1444.

AUTHOR PHOTOS AND BIOGRAPHY

Hengxing Cai received his B.S. degree in Traffic Engineering from Hainan University, Haikou, China in 2015.He is currently pursuing the M.S. degree majoring in Intelligent Transportation Engineering in Sun Yat-sen University, Guangzhou, China. His research interests include traffic data mining, traffic safety, and intelligent transportation systems.

Renxin Zhong received the B.Eng in Electronic Engineering from Sun Yat-sen University, the M.Phil in Automatic Control Engineering from The Chinese University of Hong Kong, the Ph.D. degree in Transportation Engineering from The Hong Kong Polytechnic University in 2005, 2007, and 2011, respectively. He joined the Research Center of Intelligent Transportation Systems of Sun Yat-sen University as an Associate Professor in 2012. His main research interests include dynamic traffic surveillance and assignment; traffic incident detection and management strategies; machine learning and data mining for transportation big data analysis; optimal and nonlinear control theory with applications in transportation engineering.