Article

Location Extraction and Prediction Method Based on Floating Car Spatial-Temporal Trajectory

Shaoming Pan , Ziying Li and Yanwen Chong *

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing,
Wuhan University, Wuhan 430079, China; pansm@whu.edu.cn (S.P.); zy.lili@whu.edu.cn (Z.L.)

* Correspondence: ywchong@whu.edu.cn

Received: 30 March 2020; Accepted: 4 May 2020; Published: 7 May 2020

Abstract: Predicting the next important location by mining the user's historical spatial-temporal trajectory can be done for behavioral analysis and path planning. Since extracting the important location precisely is the premise of next location prediction, an enhanced location extraction algorithm is proposed to meet the requirements of dynamic trajectory via dynamic parameter estimation. To realize the estimation of parameters dynamically, the differences of floating car velocity in terms of spatial distribution and behavior in time distribution are considered in the location extraction algorithm. Then, an improved recurrent neural network (RNN) model is designed to mine the variation law of floating car trajectories to improve the accuracy of important location prediction under different conditions. Different from the traditional prediction model considering only the constraint of distance, the attention mechanism and semantic information are considered simultaneously by the proposed prediction model. Finally, the floating car trajectory of Beijing is selected for our experiments, and the results show that the proposed location extraction algorithm can meet the requirements of a dynamic environment and our model achieves high prediction accuracy.

Keywords: location extraction and prediction; spatial-temporal distribution; dynamic location extraction; RNN; self-attention

1. Introduction

With the popularity of smartphones and other equipment that can be used to record locations via the Global Navigation Satellite System (GNSS) [1], capturing and recording trajectory data based on such equipment becomes more convenient. Thus, much research on location-based service (LBS) has been proposed to improve service capabilities by mining the laws from recorded trajectory data, such as for targeted advertising, road assistance and navigation, personnel tracking, point-of-interest (POI) recommendations [2], and so on. Obviously, a complete LBS system can not only obtain a user's current location, but also predict the next possible location based on their historical trajectory information, so as to plan the navigation path in advance or analyze the user's behavior. Since recording and obtaining trajectory data have become easy and convenient, and predicting the next location based on historic behavior to find future destinations is a key factor for complete LBS [2], so it is important to accurately predict the next location.

In fact, the research indicates that people's mobility is highly dependent on their historical behaviors [3]. Location prediction can be divided into two types: personalized and popularized [4]. Personalized location prediction mainly looks at the historical trajectory of a single user, while popularized location prediction mines the travel habits of many users. Since personalized location prediction can be used to provide guidance for navigation and personality recommendations, it plays an important role in the research of location prediction, so this paper also pays attention to that research. It is clear that the trajectory of different users is basically different even if their visited locations might

be similar. Since the personalized location prediction is based on user's own behaviors, building a separate prediction model for each user might be preferable [5]. In addition, the different characteristics of user behavior have different influences on prediction. For example, predicting dynamic behavior is more difficult than predicting fixed behavior, and predicting many possible destinations is also more difficult than predicting only several possible destinations. Much different from pedestrian, bus, subway, train, and airplane travel, where the behavior features are mostly fixed or the destinations are mostly limited, not only are the behaviors of floating cars dynamic, but also their destinations are hugely variable. Thus, research based on trajectory data recorded in floating cars is more meaningful and more challenging.

Moreover, a location prediction algorithm mainly predicts the user's next important location, such as a mall, school, important road intersection, residential area, or attraction. In order to make meaningful predictions, discarding as much of the unimportant data as possible and keeping the most important data is a priority due to the massive and redundant location data recorded by floating cars. Thus, it is necessary to propose an important location extraction algorithm to filter and cluster the original trajectory data in order to reduce the number of trajectory points and obtain important locations as the input of the prediction model.

Based on the above analysis, an important location extraction algorithm and a precise location prediction model are the two aspects of location prediction research for LBS, the former for reducing the size of the dataset and data redundancy, and the latter for precise prediction of behaviors.

In order to obtain important locations for the prediction model, many traditional location extraction methods can be used, such as manual tagging algorithms [2,6] and clustering methods [7–9]. Among them, attention-based spatiotemporal gated recurrent unit (ATST-GRU) introduced a method to realize POI recommendations by users sharing locations and check-in information with others in location-based social networks (LBSNs) [2]. Tree-based hierarchical graph (TBHG) also introduced a method that marks important locations by matching the trajectory with the map [6]. It is clear that both of these methods require geographical information, which will lead to heavy workloads. Furthermore, it is difficult to obtain the check-in information of users because of personal privacy issues.

Hybrid methods and clustering algorithms are two typical models that can be used to extract locations and avoid personal privacy issues. A typical hybrid method simply annotates POIs with meaningful information from LBSNs [10]. Since an important location represents a geographic area where a user stayed for a certain interval of time, extracting the important location based on two scale parameters, time, and distance, instead of collecting check-in information can also avoid personal privacy issues. Therefore, many clustering algorithms have been proposed to extract locations by selecting appropriate distance and time thresholds [11], such as density-based spatial clustering of applications with noise (DBSCAN) [7–9], K-means [12], and ordering points to identify the clustering structure (OPTICS) [13] algorithms.

Although the above methods can be used to obtain satisfactory results in location extraction, artificially configuring the threshold and parameters, which have a great influence on the clustering results, is their obvious disadvantage. Due to the dynamic behaviors of different floating cars, obtaining common parameters for different users is impossible. Thus, an adaptive algorithm that can meet the dynamic requirements of different users should to be considered. In this paper, a dynamic DBSCAN (D-DBSCAN) algorithm based on the traditional DBSCAN algorithm is proposed to dynamically adjust the value of Eps (the neighborhood radius) in order to extract important locations via different parameters based on different situations. The D-DBSCAN algorithm considers the dynamic information of trajectories and can effectively filter strip clusters. This has been proved by location extraction experiments.

As the traditional domain for trajectory analysis, location prediction based on floating car trajectory data was a kind of trip-matching process in early approaches [14]. Sub-trajectory synthesis (SubSyn) [14] divided the trajectories into sub-trajectories and then fused the sub-trajectories of different users to match current trips. Different from the above trip-matching algorithm, various Markov chain

models offered a kind of destination-matching method where possible next locations were collected in advance and the prediction result was the probability of different important locations. Among them, Ashbrook et al. proposed a prediction method using the Markov model, where each node was an important location clustered from Global Positioning System (GPS) data, and a transition between two nodes represented the probability of the user traveling between those two important locations [7]. On this basis, Simmons et al. [15] proposed a hidden Markov model (HMM) to predict destinations. Furthermore, Ashbrook et al. [16] built models by adding the concept of support points to the hidden Markov model to improve prediction performance. Although the experimental results showed that the strategy of support points can be used to improve the performance of the algorithm, the model still cannot achieve high accuracy because a different choice of strategy will invariably lead to different performance, and no appropriate method has been proposed to match the various requirements in different situations. It is possible that a passenger may go to a totally different place where the floating car has never been. Thus, not only trip-matching algorithms but also destination-matching methods cannot meet the requirements of new next-location prediction. Furthermore, the above approaches will be ineffective when there is an absence of prior knowledge.

With the development of machine learning, deep learning technology has been widely used for next-location prediction. First, a kind of Bayesian network model was designed, and descriptive information of the trajectory was added to the model as a feature to perform destination prediction [14]. In addition, a location-prediction algorithm mainly makes predictions based on contextual information of the trajectory and the current location. Since the historical trajectory information is typical time series data, the location-prediction algorithm is also a typical time series prediction process. Recently, recurrent neural networks (RNNs) [17] have been adopted in machine translation [18], target recognition [19], video behavior recognition [20], sentiment classification, and image caption generation [21] and show promising performance in processing time series prediction compared with traditional methods. Therefore, RNNs can be used to predict next important locations [22,23]. Liu and co-workers [22] and Al-Molegi and colleagues [23] focused on using a set of features to obtain a good prediction performance. In these models all historical trajectory points have the same importance whether they are located in an intersection or in the middle of a straight road. In fact, trajectory points located in intersections, which greatly affect the turning direction, may play a more important role in next-location prediction. Some attention mechanisms need to be further considered for prediction algorithms based on the traditional RNN model in order to track the dramatic changes of floating car trajectories.

To remedy the two problems, this paper proposes a D-DBSCAN location extraction algorithm to cluster important locations and an attention-RNN location prediction algorithm to predict the next location. The main contributions are summarized as follows:

- (1). A novel dynamic important location extraction algorithm based on DBSCAN is proposed to extract important locations via different parameters based on different situations to meet the requirements of dynamic situations. The algorithm can dynamically adjust the parameters by tracking different user behaviors and can effectively filter out strip shape clusters in order to avoid using such invalid data as input for the prediction algorithm.
- (2). We propose attention-RNN location prediction, which can assign a different level of attention to historical track points to grasp the spatial characteristics of trajectories and closely track the user trajectory.
- (3). A time-step window mechanism is added to the attention-RNN model to reduce the time consumption and computational complexity.

The rest of this paper is organized as follows. We briefly introduce some related work on important location extraction and prediction models in Section 2. The dynamic important location extraction algorithm (D-DBSCAN) and attention-RNN location prediction model are proposed in Section 3. Section 4 reports the experimental design, performance metrics, and extensive experimental results and discusses the merits and drawbacks of our results and the baseline methods (RNN [17], space

time features based RNN (STF-RNN) [23], and spatial-temporal RNN (ST-RNN) [22]). Finally, the conclusions are drawn in the last section.

2. Related Works

The DBSCAN algorithm does not need to set a fixed number of clusters in advance and has advantages over other algorithms, and the clustering results are not constrained by the cluster shape. However, the traditional algorithm still has some disadvantages, including that the parameters of the neighborhood radius Eps and the minimum number of points contained in the neighborhood ($Minpts$) must be set artificially, and the selection of parameters has a crucial effect on clustering results [8]. Unfortunately, it is difficult to select an appropriate value of Eps for the traditional DBSCAN algorithm, especially in massive nonuniform distribution conditions. For example, some important locations may be merged, or a single important clustering location may be split into many different clusters. In order to solve the parameter setting problem of the traditional DBSCAN model, Zhou et al. proposed a DBSCAN algorithm based on data partitioning [24]. The algorithm partitions the data according to density and establishes an R^* tree in each region to obtain a K -dist map. According to the K -dist map, the appropriate values are selected in different partitions, and finally the clustering results of each region are combined to obtain the final results.

The partition-based DBSCAN algorithm can be used to obtain clustering locations that are not constrained by shapes, and small areas can also be preserved so as to avoid discarding some important location information. However, due to the massive dataset, the partition-based DBSCAN algorithm can obtain many strip clusters that are formed by points on the road. Most of these strip clusters are meaningless locations and should not be input into the prediction model. On the other hand, some clusters that are formed by congestion points and intersection points play an important role in the location prediction application and these clusters need to be retained.

Based on the above analysis, a dynamic DBSCAN (D-DBSCAN) algorithm is necessary to dynamically adjust the value of Eps and extract important locations via different parameters in different situations, such as different velocity and time information, rather than artificially setting the parameters. The details of the D-DBSCAN algorithm are presented in Section 3.

In order to deal with the problem of trajectory location prediction, Brébisson et al. [5] inputted trajectory points one by one into an RNN network and used the memory function of the hidden layer to achieve the prediction purpose. Liu et al. extended the RNN and proposed a novel method called spatial temporal RNN (ST-RNN) [22]. ST-RNN can model local temporal and spatial contexts in each layer with time-specific transition matrices for different time intervals and distance-specific transition matrices for different geographical distances [22]. Al-Molegi et al. [23] proposed a method to leverage RNN to model people's movement behaviors in order to predict their next location. Space and time are included in the network as features, where their internal representations are learned by the network itself rather than relying on a manmade representation.

In the above algorithms, all historical trajectory points have the same importance whether they are located in an intersection or in the middle of a straight road. In fact, trajectory points located in intersections, which greatly affect the turning direction, may play a more important role in next-location prediction. In order to fully consider the different influence weights of historical trajectory data on predicted locations, an attention mechanism module is added in the traditional RNN network in this paper. At the same time, in order to solve the problem that the performance of the model will deteriorate rapidly as the length of the input sequence increases, a time-step window should be considered to reduce the time consumption and computational complexity and improve training efficiency. In addition, an attention mechanism and the trajectory semantic information must be fully considered to grasp the spatial characteristics of trajectories.

3. Methodology

3.1. Dynamic Location Extraction Method

The D-DBSCAN algorithm integrates the track point velocity information dynamically, which can adjust the key parameters of the algorithm dynamically. The main principles of the D-DBSCAN algorithm are as follows: (1) it calculates the instantaneous velocity of each trajectory point according to time sequences, (2) divides the points into different regions based on their velocity so as to reduce the velocity difference in the same region to better set Eps , and (3) it takes the velocity of the track point as an assist, dynamically adjusting the value of Eps .

Generally, denote $P = \{p_1(x_1, y_1), p_2(x_2, y_2), \dots, p_n(x_n, y_n)\}$ and $T = \{t_1, t_2, \dots, t_n\}$ as the trajectory data and the corresponding timestamp data of a certain user, then the velocity of each trajectory point $V = \{v_1, v_2, \dots, v_n\}$ can be computed based on P and T . By sorting V in descending order to obtain the sorted velocity $V' = \{v'_1, v'_2, \dots, v'_n\}$ and then subtracting the sorted velocity one by one to get the velocity differences $\Delta V = \{\Delta v_1, \Delta v_2, \dots, \Delta v_{n-1}\}$, the maximum velocity $v_{max} = v'_1$ and the minimum velocity $v_{min} = v'_n$ can be easily found from V' . After that the maximum $N - 1$ velocity differences $\Delta V' = \{\Delta v'_1, \Delta v'_2, \dots, \Delta v'_{N-1}\}$ is calculated ΔV , where all elements of $\Delta V'$ are more than a minimum velocity difference threshold ω . It is clear that the different values of ω may lead to obtain different numbers of the maximum velocity differences.

In order to divide the points into different regions, velocity partition thresholds can be computed firstly based on the velocity differences $\Delta V'$. $\Delta v'_i$ is obtained by subtracting v'_m and v'_{m+1} , and without loss of generality, denotes the smaller (v'_{m+1}) of two corresponding velocities (v'_m and v'_{m+1}) of each maximum velocity differences $\Delta v'_i$ as a velocity threshold, then the corresponding $N - 1$ velocity thresholds of the maximum $N - 1$ velocity differences can be noted as $\{vt_1, vt_2, \dots, vt_{N-1}\}$, and thus, by sorting VT in ascending order to obtain the sorted $VT = \{vt_0, vt_1, vt_2, \dots, vt_{N-1}, vt_N\}$. VT can be defined as velocity partition thresholds, where $vt_0 = v_{min}$ and $vt_N = v_{max}$. Then, all trajectory points can be easily divided into N regions $R = \{R_1, R_2, \dots, R_N\}$ based on VT . Thus, the mean velocity of each region $\bar{V} = \{\bar{v}_1, \bar{v}_2, \dots, \bar{v}_N\}$ can be computed.

In order to clearly demonstrate the above process to obtain velocity partition thresholds, a simple example is shown in Table 1. It is also clear that the different number of maximum velocity differences may lead to obtaining a different number of regions.

Table 1. Example of computing velocity partition thresholds.

Parameters	Value
V	$\{20, 80, 96, 100, 98, 60, 10, 8, 3 m/s\}$
V'	$\{100, 98, 96, 80, 60, 20, 10, 8, 3 m/s\}$ and $v_{max} = 100$ m/s, $v_{min} = 3$ m/s
ΔV	$\{2, 2, 16, 20, 40, 10, 2, 5 m/s\}$
$\Delta V'$	$\{16, 20, 40, 10 m/s\}$ when assuming $\omega = 6(m/s)$ and all elements greater than 6 are selected from ΔV as the elements of $\Delta V'$.
VT	$\{v_{min}, 10, 20, 60, 80, v_{max} m/s\} = \{3, 10, 20, 60, 80, 100, m/s\}$. Since $\Delta v'_4 = 10$ is obtained via $v'_6 - v'_7 = 20 - 10$, the smaller of two corresponding velocities $v'_7 = 10$ is then selected as the one element of VT .

Obviously, the velocities of the different trajectory points V which belong to the same region are similar and thus the same parameter of Eps can be used. Therefore, the parameter Eps of each region can be dynamically calculated by Equation (1), and then clustering in each region. Finally, the clustering results of each region are combined to obtain the final results. The complete procedure is described as Algorithm 1:

$$Eps_i = \varepsilon \frac{1}{\bar{v}_i}, i = 1, 2, \dots, N \quad (1)$$

where ε is the influence factor which can be simply estimated via calculating the K -dist map [24] (the appropriate value of ε is selected as 0.06 in our experiment). Moreover, although the velocity based

partition method can reduce the velocity difference of the trajectory points in the same region so as to more appropriately set Eps parameter for each region, partitioning too many regions based on a too small velocity difference threshold ω may lead to splitting a single important clustering location into several fragmented location parts and which will reduce the accuracy of important location clustering as well as increase computational complexity. By the results of an ablation experiment, $\omega = 3(\text{m/s})$ is defined as the minimum velocity difference threshold in our experiment so as to obtain the optimal region number N via the velocity differences ΔV .

Algorithm 1: D-DBSCAN algorithm for location extraction

Input: Given the trajectory data P , and the timestamp data T .

(1) Calculate V based on P and T , and sort V in descending order to get V' .

(2) Obtain v_{max} and v_{min} from V' and subtract V' one by one to get ΔV .

(3) Get the velocity partition thresholds VT based on ΔV , ω , v_{max} , and v_{min} .

(4) Partition all trajectory points into N regions R based on VT :

For $v_i \in V$

 If $vt_0 \leq v_i \leq vt_1$

$v_i \in R_1$

 else if $vt_1 < v_i \leq vt_2$

$v_i \in R_2$

 ...

 else if $vt_i < v_i \leq vt_{i+1}$

$v_i \in R_{i+1}$

 ...

 else if $vt_{N-1} < v_i \leq vt_N$

$v_i \in R_N$

 End if

End for

(5) Calculate the mean velocity of each region $\bar{V}$.

(6) Update Eps via Equation (1) for each region, and cluster by region.

(7) Combine the clustering results of each region and calculate the latitude and longitude of the center point of each cluster.

Output: Location extraction results (sequence of locations).

As shown in Algorithm 1, a larger instantaneous velocity of the trajectory point indicates a smaller Eps based on Equation (1). Thus, the D-DBSCAN algorithm can realize dynamic clustering under different velocity conditions by adjusting Eps dynamically, and then key locations, such as intersection points, can be retained and strip clusters can be filtered out, and finally a better location extraction result can be obtained to provide a better guarantee for location prediction. Moreover, the algorithm takes full account of the dynamic characteristics of trajectories, thus better location extraction results can be obtained.

3.2. Attention-RNN Location Prediction

3.2.1. Traditional RNN Prediction Network

The architecture of an RNN is a recurrent structure, and traditional RNN includes the Elman network and the Jordan network. The Elman network feeds the hidden layer (h_t) back into the recurrent structure, while the Jordan network feeds the output of the network (o_t) back into the recurrent structure. Many network variants are derived from the Elman network, so in general, when it comes to RNN, it refers to the Elman network. The Elman network is adopted in this paper, and the hidden layer (h_t) is fed back into the recurrent structure. At each time t , we can predict the hidden layer (h_t) by

the previous moment hidden layer (h_{t-1}), and then feed the new hidden layer (h_t) back into the next hidden status. The formulation of the hidden layer in RNN is:

$$h_t = g(W^{(x)}x_t + W^{(h)}h_{t-1}) \quad (2)$$

where the activation function $g(\bullet)$ is a $\tanh$ function, $W^{(x)}$ is the input layer weight, and $W^{(h)}$ is the hidden layer weight.

The final result o_t of the network can be obtained by using the appropriate activation function $\sigma(\bullet)$ and the output layer weight $W^{(o)}$ on the hidden layer state h_t generated from the hidden layer.

$$y = o_t = \sigma(W^{(o)}h_t). \quad (3)$$

3.2.2. Attention-RNN Location Prediction

Due to the unequal time interval from the predicted location and different spatial adjacencies, different historical trajectory data have unequal influence weights on the predicted location. For example, the next location is greatly affected by the trajectory point at that turning point when the direction of the track changes. The attention mechanism module is added to the traditional RNN network to fully reflect the different influence weights of the historical trajectory data on the predicted location. The architecture of the proposed location prediction network in this paper is represented in Figure 1.

Figure 1. Attention recurrent neural network (RNN) structure.

As shown in Figure 1, a time-step window model is designed to divide the original trajectory sequence into several sub-tracks to reduce the amount of data and improve the training efficiency of the network, and the trajectory semantic information (SI) is extracted by the embedding layer. The attention coefficient of the proposed self-attention mechanism (SAM) can be calculated via azimuth information, which can also be obtained from the original trajectory sequence.

After that, a feature vector of the network input layer can be obtained by concatenating the sub-tracks with the trajectory description information, which goes through the embedding layer, and

the attention coefficients are added into the RNN network; also, the attention mechanism is adopted to make the network better grasp the spatial characteristics of trajectories. Finally, the prediction location (latitude and longitude) is obtained by the activation function. The different parts of the attention-RNN location prediction method are described in detail as follows:

(a) Time-step window: In location prediction experiments, padding based on the length of the longest track is usually used to solve the problem of unequal track sequence lengths, then track points are inputted into the RNN one by one. However, because some of the tracks are collected by the user for 24 hours or longer, padding all tracks will result in a sharp increase in data volume, and it turns out that the influence of historical trajectory points on predicted location gradually weakens over time. If the training is not controlled, it will not only be a waste of meaningless resources, but also will likely be counterproductive to the prediction results. Inspired by the concept of mask convolution in pixel convolutional neural network (CNN) [25], a time-step window is added to divide the original trajectory into several sub-tracks so that all tracks are guaranteed to have the same number of timestamps as the window size and the scope of the historical information is controlled in a certain range. The experimental results show that adopting an appropriate time-step window size can improve not only the efficiency of network training but also the prediction accuracy.

(b) Trajectory semantic information: Using only the latitude and longitude of the GPS point for location prediction, the network can learn poorly because the feature dimension is too small, and the accuracy of the predicted result is not sufficient. In the location prediction experiment, not only the latitude and longitude coordinates of the trajectory data but also some description information should be utilized. Considering that there are certain differences in people's travel destinations between weekends and workdays, an embedding layer is added to the network to mine deeper semantic information (SI) from the data, and to determine whether the trajectory occurs on a weekend or weekday. After the embedding layer, the time information is concatenated with the trajectory sequence, and then input into the RNN.

(c) Self-attention mechanism: In order to fully reflect the different influence weights of the historical trajectory data on the predicted location, this paper introduces a self-attention mechanism (SAM) to learn a set of weight information based on the azimuth changes produced by historical trajectories. In the process of training, each historical track point is assigned a different level of attention.

In the process of obtaining the attention coefficient, the concept of coordinate azimuth is introduced [26]. Usually the coordinate azimuths of the trajectory data $P = (p_1, p_2, \dots, p_n)$ are defined as $A = (a_1, a_2, \dots, a_n)$. The azimuth of trajectory point p_i is a_i , and azimuth a_i is the angle formed by clockwise rotation from the north to the line connected by track point p_i and its previous point p_{i-1} based on the definition of azimuth. Thus, the azimuths of trajectory P are as follows:

$$A = (a_1, a_2, \dots, a_n | 0 \leq a_i \leq 2\pi) \quad (4)$$

where $a_1 = 0$, and a_i is the azimuth of p_i . Then the angle of the entire track changes can be denoted as:

$$\beta = (\beta_1, \beta_2, \dots, \beta_n) = (a_1, |a_2 - a_1|, |a_3 - a_2|, \dots, |a_n - a_{n-1}|) \quad (5)$$

When β_i is taken as π , the track direction changes the most, and the degree of change is symmetrically distributed from π to 0 and from π to 2π . Therefore, the correspondence between attention coefficients s and β is:

$$s_i = \delta \sin\left(\frac{\beta_i}{2}\right) \quad (6)$$

where δ is a scaling factor, and the network's powerful self-learning ability is used to adjust the attention coefficient and obtain $S = (s_1, s_2, \dots, s_n)$. Using the attention factor obtained and the attention layer weight $W^{(s)}$, Equation (3) is modified to update the hidden layer:

$$h_t = g(W^{(s)}S + W^{(x)}x_t + W^{(h)}h_{t-1}). \quad (7)$$

4. Experimental Results and Analyses

4.1. Dataset

The GPS trajectory dataset used in this paper was collected by the Microsoft Asia Research Institute Geolife project [27,28] from April 2007 to August 2012, including trajectory data of 182 users. There are 73 labels of the trajectories according to transportation mode such as driving, taking a bus, riding a bike, or walking. The total distance and duration of different transportation modes are listed in Table 2.

Table 2. Total distance and duration of transportation modes.

Transportation Mode	Distance (km)	Duration (h)
Walk	10,123	5460
Bike	6495	2410
Bus	20,281	1507
Car and taxi	32,866	2384
Train	36,253	745
Airplane	24,789	40
Other	9493	404
Total	140,304	12,953

Data collection duration of different users varies, and the specific distribution is shown in Figure 2.

Figure 2. Duration of user data collection.

As shown in Figure 2, according to the collection duration, the experimental data can be classified into three categories: users with a long collection duration (more than 1 month), users with a medium collection duration (1 week to 1 month), and users with a short collection duration (less than 1 week). In order to verify the adaptability of the model under different trajectory lengths, 30% of users in each category were randomly selected for the experimental data. Eventually, experimental results of each category are listed.

4.2. Data Preprocessing

The quality of GPS data is affected by sensors, transmission paths, etc., and data leakage, data loss, and other noise pollution are prone to occur, which will adversely affect subsequent data analysis. Therefore, noise point filtering is required before trajectory prediction. Although the user trajectories in the Geolife dataset are widely distributed, most of them were collected in Beijing, China. Before the conventional noise processing, the latitude and longitude range (115.70° N–117.37° N, 39.40° E–41.03° E) was used to select the data inside the study area. Only floating car trajectory data according to transportation mode were selected according to the analysis in Section 1.

Several typical preprocessing methods for GPS data noise were selected: outlier filtering, smoothing, and data interpolation. The data sampling interval used in the experiment was small and the gross error in the data had the largest influence on the trajectory prediction. Zhang et al. [29] compared several filtering methods under several different types of noise. The results showed that

median filtering works better when dealing with gross errors, so median filtering was used for data filtering.

4.3. Evaluation Metrics

In order to evaluate the validity of the proposed location extraction method, the silhouette coefficient (SC) was used [30]. The SC is a metric that does not need to know the true labeling of the dataset. The value of SC ranges from -1 to $+1$, and a high SC value indicates a good location extraction result.

The haversine distance between two points was used to measure the error between predicted result $\hat{y}$ (longitude and latitude of a predicted location) and true value y (longitude and latitude of a true destination location). The haversine distance between point $x(lon_x, lat_x)$ and point $y(lon_y, lat_y)$ is:

$$d_{haversine}(x, y) = 2R \cdot \arctan\left(\frac{f(x, y)}{f(x, y) - 1}\right) \quad (8)$$

where R is the radius of the Earth, and $f(x, y)$ is defined as:

$$f(x, y) = \sin^2\left(\frac{lat_y - lat_x}{2}\right) + \cos(lat_x) \cos(lat_y) \sin^2\left(\frac{lon_y - lon_x}{2}\right) \quad (9)$$

In this paper, mean absolute error (MAE) and root mean square error (RMSE) are used to measure the experimental results of the prediction algorithm. MAE is the average haversine distance, which can be calculated as:

$$MAE = \frac{1}{n} \sqrt{\sum_{i=1}^n d_{haversine}(y, \hat{y})} \quad (10)$$

RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n d_{haversine}(y, \hat{y})^2} \quad (11)$$

For the sake of simplicity, all experimental results in this paper are expressed as integers.

4.4. Location Extraction Experimental Results

In order to evaluate the precision of the proposed D-DBSCAN algorithm compared with the traditional DBSCAN, the experiment of important location extraction is demonstrated with a randomly selected user's trajectory, and the results of different clustering algorithms can be visualized in Figure 3.

Figure 3. Location extraction results of: (a) density-based spatial clustering of applications with noise (DBSCAN) and (b) dynamic DBSCAN (D-DBSCAN) algorithms.

As shown in Figure 3, the proposed D-DBSCAN method effectively filters out strip shape clusters and preserves some small clusters. It is found that the proposed method can be used to dynamically adjust the clustering parameters, so it can take a relatively small Eps to filter out strip shape clusters when the user is driving on the road at high velocity. As shown in Figure 3a,b, the D-DBSCAN algorithm can obtain better performance in filtering strip shape clusters than DBSCAN, which can reduce disturbance to the prediction model and help to improve the precision. Furthermore, the parameters can also be dynamically adjusted to take a relatively large value when the velocity is low to preserve these important location points (e.g., points A, C and B, D in Figure 3b, where A and C are intersections and B and D are congestion points in the trajectory).

Accuracy comparisons can be found in Table 3, which provides the numbers of track points during clustering and obtained locations based on both algorithms.

Table 3. Location extraction results of different clustering algorithms. SC, silhouette coefficient.

Method	Total Track Points	Number of Clusters	Number of Locations	SC
DBSCAN	592,912	189,180	383	−0.429
D-DBSCAN	592,912	104,312	1093	0.113

As the results shown in Table 3, from the evaluation metric perspective, D-DBSCAN has a higher SC (last row) than DBSCAN. From the perspective of the needs of the prediction model, similar to the previous analysis, D-DBSCAN can effectively filter strip shape clusters, so a lot of unimportant data are discarded and only the most important clusters are preserved, thus, as shown in Table 3 (second row), fewer track points remain with the D-DBSCAN clustering algorithm. At the same time, erroneously merging some different important locations due to data nonuniform distribution are effectively avoided based on the dynamic parameters' strategy, and more important locations are more accurately obtained by the D-DBSCAN algorithm than DBSCAN, which can also be seen in Table 3 (third row).

It is obvious that discarding as many unimportant track points as possible can reduce calculation cost; moreover, preserving the important locations and dividing locations into as many different clusters as possible are two key aspects of providing sufficient input data for the prediction model and avoiding disturbance to the model. The results show that the proposed D-DBSCAN can achieve the best performance in both aspects.

Furthermore, a series of velocity differences threshold ω were used in the experiment so as to check the influence of different minimum velocity difference threshold ω to the result of important location clustering, and the comparison experimental results are shown in Figure 4.

Figure 4. Comparison experimental results with different minimum velocity difference thresholds.

According to the analysis above, partitioning trajectory points into different regions with an optimal minimum velocity difference threshold ω is the key to both obtaining a velocity similarity in the same region and avoiding the incorrect extracting important positions by appropriately setting Eps for each region. It can be seen from Figure 4 that the SC value gradually increases as the value of the minimum velocity difference threshold increases from 1(m/s) to 3(m/s). However, the SC value contrarily decreases as the value of the minimum velocity difference threshold continuously increases from 3(m/s) to 10(m/s). Since the value SC indicates the accuracy of important location clustering, $\omega = 3(\text{m/s})$ is the optimal minimum velocity difference threshold in our experiments.

4.5. Location Prediction Experiment Results

Users have different behaviors and interests in different time periods and regions. According to the collection duration, the experimental data can be classified into three categories. In order to verify the prediction ability and adaptability of the proposed algorithm, 30% of users in each category were randomly selected for the location prediction experiment. The average accuracy of each category is listed. The size of time-step window is 10 in our experiment.

We compared our model with some baseline approaches: RNN [17], STF-RNN [23], and ST-RNN [22]. The results are shown in Table 4.

Table 4. Comparison of experimental results between users. MAE, mean absolute error; RMSE, root mean square error; STF-RNN, space time features based RNN; ST-RNN, spatial temporal RNN.

Methods	Long Collection Duration Users		Medium Collection Duration Users		Short Collection Duration Users		Mean	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
RNN	1110	2502	1591	3772	1685	1889	1462	3621
STF-RNN	471	788	562	1509	264	623	284	973
ST-RNN	164	453	265	1164	152	363	194	660
Ours	61	312	172	789	150	287	128	463

It can be seen from the results in Table 4 that the proposed method is superior to the others in trajectory prediction based on users with historical trajectories of various lengths. In this paper, the attention mechanism and trajectory semantic information are added into the RNN model. Therefore, the prediction accuracy has an average increase of 87% compared to RNN. The more effective time-specific transition matrices and distance-specific ones are adopted in ST-RNN to extract temporal and spatial information, which makes it have higher prediction accuracy than STF-RNN. Since more semantic information is used in the proposed method and the attention mechanism fully reflects the different influence weights of the historical trajectory data on the predicted location, the prediction accuracy has an average increase of 55% compared to STF-RNN and has an average increase of 34% compared to ST-RNN.

4.6. Ablation Experiment Results

The size of the time-step window determines the size of the history track reference range. The larger the step window, the more historical track data is needed; the prediction accuracy will be affected accordingly, and the model training time will also become longer. In order to prove that applying an appropriate time-step window can reduce prediction error, different time-step windows were used in the experiment. The comparison experimental results with different time-step windows are shown in Figure 5.

Figure 5. Comparison experimental results with different time-step windows.

It can be seen from Figure 5 that the error of the prediction network is decreased sharply, and the prediction accuracy is increased when the time step is changed from 50 to 10. In addition, reducing the size of the time-step window can contribute to a reduction in the amount of data during network training and then reduce training consumption. When the time step is changed from 10 to 5, although the time consumption is reduced, the prediction error also increases, and the accuracy of the prediction model drops sharply. The reason for this result is that when the time step is 5, the network has a small history track reference range and there is too little historical information to learn the user's travel habits. It can be concluded that applying a suitable time-step window size can not only improve accuracy but also reduce time consumption.

Furthermore, in order to observe the contribution of each part of the model, an ablation experiment was conducted, and the experimental result is shown in Table 5, where RNN is the base model and SI and SAM represent the semantic information strategy and self-attention mechanism model, respectively.

Table 5. Results of ablation experiment. SI, semantic information; SAM, self-attention mechanism.

Method	Long Collection Duration Users		Medium Collection Duration Users		Short Collection Duration Users		Mean	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
RNN	1110	2502	1591	3772	1685	1889	1462	2721
RNN + SI	269	566	325	862	517	605	370	678
RNN + SI + SAM	61	312	172	789	150	287	128	463

It can be seen from the results in Table 5 that each part of the proposed model plays its due role in improving prediction accuracy based on users with historical trajectories of various lengths. Since most users travel between their fixed residences and workplaces on workdays but are free to go anywhere they want on weekends, their behaviors are quite different between weekends and workdays. Thus, deeper semantic information from the data can better match the interest points of users to improve prediction performance. As the results in Table 5 show, the prediction accuracy of all

different collection duration users is significantly improved after adding semantic information, and the average prediction accuracy is improved by 75%. Moreover, the directions of some key locations, such as the road intersections of the track, may play an important role in deciding whether users go to work or back home. Thus, paying close attention to the changes of track direction is another key factor to follow users' behaviors, so the additional SAM strategy can be used to further improve the accuracy of prediction. It can be seen from Table 5 that the average prediction accuracy is improved by 64% after applying self-attention mechanism.

5. Conclusions

In this paper, a novel dynamic important location extraction algorithm based on DBSCAN is proposed to meet the requirements of dynamic situations. Visualizing the clustering results, we can see that a better location extraction result is obtained. In addition, in the prediction model, a time-step window is added based on the RNN to shorten the training time and reduce the equipment requirements, and a self-attention mechanism and trajectory semantic information are added to the RNN model. It is proven by experiments that using a suitable time-step window size can not only improve accuracy but also reduce time consumption and adding a self-attention mechanism and trajectory semantic information can improve prediction accuracy. In the future, extracting more rich semantic information from the trajectory data to improve the adaptive ability of the model and realize the prediction of unknown important points is a possible research direction. Furthermore, location information belongs to the user's private information, so starting simultaneously collaborative research on location prediction and privacy protection is necessary and meaningful.

Author Contributions: Conceptualization, Shaoming Pan, Ziyang Li, and Yanwen Chong; methodology Shaoming Pan, Ziyang Li, and Yanwen Chong; software, Shaoming Pan, Ziyang Li, and Yanwen Chong; validation, Shaoming Pan, Ziyang Li, and Yanwen Chong; formal analysis, Shaoming Pan, Ziyang Li, and Yanwen Chong; investigation, Shaoming Pan, Ziyang Li, and Yanwen Chong; resources, Shaoming Pan, Yanwen Chong; data curation, Shaoming Pan, Ziyang Li, and Yanwen Chong; writing—original draft preparation, Shaoming Pan, Ziyang Li, and Yanwen Chong; writing—review and editing, Shaoming Pan, Ziyang Li, and Yanwen Chong; visualization, Shaoming Pan, Ziyang Li, and Yanwen Chong; supervision, Shaoming Pan, Ziyang Li, and Yanwen Chong; project administration, Shaoming Pan, Ziyang Li, and Yanwen Chong; funding acquisition, Shaoming Pan, Yanwen Chong. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (grant nos. 41671382, 61572372, and 41271398) and partly supported by the National Key Research and Development Program of China under grant no. 2017YFB0504202.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Camacho-Lara, S. Current and Future GNSS and Their Augmentation Systems. In *Handbook of Satellite Applications*; Springer Science and Business Media LLC: Berlin, Germany, 2013; pp. 617–654.
2. Liu, C.; Liu, J.; Wang, J.; Xu, S.; Han, H.; Chen, Y. An Attention-Based Spatiotemporal Gated Recurrent Unit Network for Point-of-Interest Recommendation. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 355. [[CrossRef](#)]
3. Lu, X.; Wetter, E.; Bharti, N.; Tatem, A.J.; Bengtsson, L. Approaching the Limit of Predictability in Human Mobility. *Sci. Rep.* **2013**, *3*, 2923. [[CrossRef](#)] [[PubMed](#)]
4. Naserian, E.; Wang, X.; Dahal, K.; Wang, Z.; Wang, Z. Personalized location prediction for group travellers from spatial-temporal trajectories. *Future Gener. Comput. Syst.* **2018**, *83*, 278–292; [[CrossRef](#)]
5. De Brébisson, A.; Simon, É.; Auvolat, A.; Vincent, P.; Bengio, Y. Artificial Neural Networks Applied to Taxi Destination Prediction. *arXiv* **2015**, arXiv:1508.00021.
6. Zheng, Y.; Zhang, L.; Xie, X.; Ma, W.-Y. Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of the 18th International Conference on Computer Systems and Technologies—CompSysTech '17, Ruse, Bulgaria, 23–24 June 2009; p. 791.
7. Ashbrook, D.; Starner, T. Using GPS to learn significant locations and predict movement across multiple users. *Pers. Ubiquitous Comput.* **2003**, *7*, 275–286. [[CrossRef](#)]

8. Ester, M.; Kriegel, H.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Knowledge Discovery and Data Mining*; Institute for Computer Science, University of Munich: Munich, Germany, 1996; pp. 226–231.
9. Zhou, C.; Frankowski, D.; Ludford, P.; Shekhar, S.; Terveen, L. Discovering personally meaningful places. *ACM Trans. Inf. Syst.* **2007**, *25*, 12. [[CrossRef](#)]
10. Zou, H.; Zhu, Z.Z.; He, X.; Zhu, A.-X. An Automatic Annotation Method for Discovering Semantic Information of Geographical Locations from Location-Based Social Networks. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 487. [[CrossRef](#)]
11. Xie, K.; Deng, K.; Zhou, X. From trajectories to activities. In Proceedings of the 2009 International Workshop on Location Based Social Networks, Seattle, WA, USA, 3 November 2009; p. 25. [[CrossRef](#)]
12. Zhou, A.W.; Yu, Y.F. The Research about Clustering Algorithm of K-Means. *Coll. Comput. Sci. Technol.* **2011**, *21*, 62–65.
13. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. *OPTICS: Ordering Points to Identify the Clustering Structure*; ACM: Philadelphia, PA, USA, 1999; pp. 49–60.
14. Xue, A.Y.; Zhang, R.; Zheng, Y.; Xie, X.; Huang, J.; Xu, Z. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, QLD, Australia, 8–12 April 2013; pp. 254–265.
15. Simmons, R.; Browning, B.; Zhang, Y.; Sadekar, V. Learning to Predict Driver Route and Destination Intent. In Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference, Toronto, ON, Canada, 17–20 September 2006; pp. 127–132.
16. Ashbrook, D.; Starner, T. Magic: A motion gesture design tool. In *Sigchi Conference on Human Factors in Computing Systems*; ACM: New York, NY, USA, 2010; pp. 2159–2168.
17. Goodfellow, I.; Bengio, Y.; Courville, A. *In Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1, pp. 367–415.
18. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078, 1724–1734.
19. Hakan, B.; Gelenbe, E. Feature-based RNN target recognition. In *Algorithms for Synthetic Aperture Radar Imagery V*; International Society for Optics and Photonics: Orlando, FL, USA, 1998.
20. Kajal, K. CARF-Net: CNN attention and RNN fusion network for video-based person reidentification. *J. Electron. Imaging* **2019**, *28*, 023036.
21. Rehab, A.; Chung, H.P.; James, H. Sequence-to-sequence image caption generator. In Proceedings of the Eleventh International Conference on Machine Vision (ICMV 2018), Munich, Germany, 1–3 November 2019. [[CrossRef](#)]
22. Liu, Q.; Wu, S.; Wang, L.; Tan, T. *Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts*; AAAI Press: Palo Alto, CA, USA, 2016.
23. Al-Molegi, A.; Jabreel, M.; Ghaleb, B. STF-RNN: Space-Time Features-based Recurrent Neural Network for Predicting People’s Next Location. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6 December 2016.
24. Zhou, S.; Zhou, A.; Cao, J. A Data-Partitioning-Based DBSCAN Algorithm. *J. Comput. Res. Dev.* **2000**, *37*, 1154–1159.
25. Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel Recurrent Neural Networks. *arXiv* **2016**, arXiv:1601.06759, 1747–1756.
26. Rutstrum, C. *The Wilderness Route Finder*; University of Minnesota Press: Minneapolis, MN, USA, 2000; ISBN 0-8166-3661-3.
27. Zheng, Y.; Li, Q.; Chen, Y.; Xie, X.; Ma, W.-Y. Understanding mobility based on GPS data. In Proceedings of the 10th International Conference on Information Integration and Web-Based Applications & Services—iiWAS ’08, Linz, Austria, 24–26 November 2008.
28. Zheng, Y.; Xie, X.; Ma, W. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. *IEEE Data Eng. Bull.* **2010**, *33*, 32–40.

29. Zhang, Z.; Zhu, J.; Kuang, C.; Ke, Y. Comparative Study and Improvement on Several De-noising Methods For Different Noise. *J. Geodesy Geodyn.* **2014**, *1*, 127–130.
30. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [[CrossRef](#)]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).