EAI Endorsed Transactions Research Article ICST.ORG

Destination Prediction by Identifying and Clustering Prominent Features from Public Trajectory Datasets

Li Yang[1], Andy Yuan Xue[1,2,*], Yuan Li[2], Rui Zhang[2]

[1]Department of Computer Science, HuBei University of Education, Wuhan, P.R. China
[2]Department of Computing and Information Systems, The University of Melbourne, Victoria, Australia

Abstract

Destination prediction is an essential task in many *location-based services* (LBS) such as providing targeted advertisements and route recommendations. Most existing solutions were *generative methods* that model the problem as a series of probabilistic events that are then used to compute the destination probability using Bayes' rule. In contrast, we propose a *discriminative method* that chooses the most prominent features found in a public trajectory dataset, clusters the trajectories into groups based on these features, and performs destination prediction queries accordingly. Our method is more concise and simple than existing methods while achieving better runtime efficiency and prediction accuracy as verified by experimental studies.

Received on 16 April 2015; accepted on 24 June 2015; published on 02 July 2015

Keywords: Trajectory Mining, Destination Prediction

doi:10.4108/sis.2.5.e2

1. Introduction

As the usage of smart phones and in-car navigation systems becomes part of our daily lives, we benefit increasingly from various types of *location-based services* (LBSs) such as route finding and location-based social networking. Destination prediction provides essential support to LBS, for example, to recommend sightseeing places, to send targeted advertisements based on destination, and to automatically set destination in navigation systems. Fig. 1 provides a schematic with the lines representing roads and the circles representing locations of interests, which may be road intersections, sightseeing places, shopping centres, etc. If one drives from l_1 to l_4, an LBS provider may predict the most probable destinations to be l_7, l_8 and l_9 based on past popular routes taken by other drivers. As a result, the LBS provider can push advertisements of products currently on sale at those locations.

In this paper, we approach the problem of destination prediction using taxi trajectories gathered in the city of Beijing as the training dataset. Following the common practice of existing work presented in the related work (Section 2), we partition the map of a city into an $n \times n$ square grid with n^2 nodes/cells. The general idea of our algorithm is as follows. We firstly convert the GPS coordinates into grid node IDs and decompose each trajectory in the trajectory dataset. For instance, a trajectory $\{n_1, n_4, n_5\}$ will be decomposed into three partially-finished trajectories $\{n_1\}$, $\{n_1, n_4\}$ and $\{n_1, n_4, n_5\}$ with the last location in these partial trajectories as the current location. We use these partial

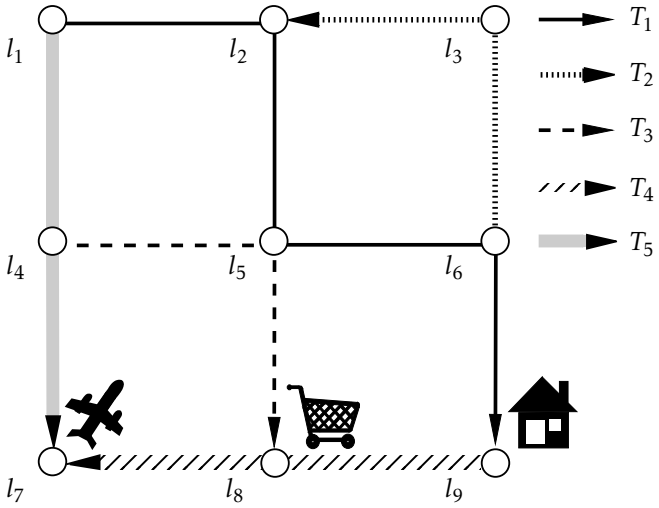

Figure 1. An example of destination prediction

*Corresponding author. Work done when visiting Hubei University of Education. Email: andy.xue@unimelb.edu.au

trajectories as the training dataset and study the properties of these trajectories to select the most prominent features such as the current location and the travel direction. Then we group the partial trajectories into different clusters based on these selected features (i.e., Trajectories with the same features are categorised to the same cluster.). Within each cluster, we compute a predicted destination by averaging the destinations for all partial trajectories in the cluster. When a query trajectory is supplied by a user, we find the cluster it belongs to according to the aforementioned features and return the pre-computed predicted destination of that cluster. We name our method *the PROFILE method* (**PRO**minent **F**eature **I**dentification and c**L**ust**E**ring).

We make the following contributions in this paper:

- We propose a novel method called *PROFILE*. PROFILE first decomposes the trajectory dataset into partially-finished trajectories, selects four features, clusters the dataset using these features, and computes a predicted destination for each cluster. Given a query trajectory with unknown destination, PROFILE returns the predicted destination of the cluster it belongs to.
- We compare the PROFILE method with a recent existing algorithm named *SubSynEA* [1]. PROFILE achieves better prediction accuracy in terms of distance deviation between the predicted destination and the true destination. This fact reflects the significance of the features that we selected.
- The PROFILE algorithm also achieves better runtime efficiency due to the simplicity of both the training and prediction phases.
- We conduct cost analysis as well as experiments to evaluate the runtime efficiency and prediction accuracy of PROFILE and SubSynEA based on a large scale real-world dataset.

Table 1 summarises the frequently used symbols.

Symbol	Explanation
$\mathbb{D}$	The historical trajectory dataset
g	Size of the map grid
m	Number of cells in a grid ($m = g^2$)
$n_i(, n_s, n_c, n_d)$	i^{th} (, starting, current, destination) cell
$d_{s\to c}$	Travelled distance between n_s and n_c
θ_c	Direction from n_c
$\theta_{s\to c}$	Direction from n_s to n_c

Table 1. Frequently Used Symbols

The remainder of the paper is organized as follows. Section 2 presents the related work. Our proposed method is described in Section 3, and its algorithms are explained in Section 4. We conduct the cost analysis in Section 5. Experimental results are presented in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

2.1. Generative and Discriminative Methods

The problem of destination prediction is to predict the destination for an ongoing trip or a *partial trajectory* based on public data. This is a typical machine learning problem, specifically a supervised learning problem. There are two categories of models that a machine learning method may use: *generative model* and *discriminative model*. Before presenting related work and our proposed method, we would like to have a discussion on these two models.

Let x and y denote the input and output variables. Discriminative model directly learns the conditional probability $P(y|x)$ or a function $y = f(x)$ from training data, whereas generative model focuses on the joint probability $P(x, y)$ first, and uses it as an intermediate result to calculate $P(y|x)$ or derive $y = f(x)$. It is a common belief that discriminative methods outperform generative methods because the joint probability $P(x, y)$ is more difficult to solve. Ng and Jordan [2] compared a generative-discriminative pair (Naive Bayes and Logistic Regression) and concluded that

> *(a) the generative model does indeed have a higher asymptotic error (as the number of training examples becomes large) than the discriminative model, but (b) the generative model may also approach its asymptotic error much faster than the discriminative model.*

Simply speaking, we should use discriminative approach when the training data is abundant and use generative method when dealing with insufficient training data. Considering the fact that our dataset contains 1.9 million taxi trajectories, we choose the discriminative approach.

2.2. Destination Prediction

As discussed above, we will use the discriminative method as it is more appropriate for the destination prediction problem. However, to our best knowledge, there is no previous work using the discriminative method to predict destinations.

Most existing methods belong to the generative method group by employing the Bayes' theorem and a grid representation of map (commonly $n \times n$ uniform grid). By using a grid, they convert the original problem of predicting a location to predicting a grid cell. By using the Bayes' theorem, they calculate the probability that a cell being the destination as

$$P(\text{dst}|T^p) = \frac{P(T^p|\text{dst})P(\text{dst})}{P(T^p)} \propto P(T^p|\text{dst})P(\text{dst}).$$

The main difference between these methods is the model built for calculating $P(T^p|\text{dst})$.

Predestination [3, 4] builds a simple model which gives larger probability to the partial trajectories towards the destination than those leaving.

$$P(T^p|\text{dst}) = \prod_{i=1}^{c} \begin{cases} p & \text{if } p_i \text{ is closer to dst than } p_{i-1} \\ 1-p & \text{otherwise} \end{cases}$$

where p is a parameter learnt from historical data.

PROCAB [5] considers vehicle movement as a Markov Decision Process and trains a cost model for every possible transition between states. This approach gives higher probability to T^p whose total cost is smaller.

$$P(T^p|\text{dst}) = \frac{\sum_{\zeta_{p_c \to \text{dst}}} e^{-\text{cost}(\zeta;\theta)}}{\sum_{\zeta_{p_s \to \text{dst}}} e^{-\text{cost}(\zeta;\theta)}}$$

where ζ denotes a possible route and θ is a parameter vector leant from historical data. SubSyn and its improved method SubSynEA [1, 6, 7] identified the data sparsity problem in destination prediction and proposed a method to address this problem. SubSynEA decomposes each trajectory in the public historical dataset into smaller segments (i.e., sub-trajectories) and combines them to generate "synthesised" trajectories. The underlying process is formulated by a second-order Markov model and the Bayesian inference framework. Other notable research outcomes in the field of prediction algorithms include [8–10].

In Section 6, we conduct experiments to compare both the efficiency and effectiveness of our proposed method against SubSyn.

3. Methodology

In this section, we present our proposed method named *PROFILE* (**PRO**minent **F**eature **I**dentification and c**L**ust**E**ring). We first partition the map of a city into an $n \times n$ grid. We convert each trajectory from a sequence of geographic coordinates into a sequence of grid nodes. Then, we decompose each trajectory in the trajectory dataset. For instance, a trajectory $\{n_1, n_4, n_5\}$ will be decomposed into three partially-finished trajectories $\{n_1\}$, $\{n_1, n_4\}$ and $\{n_1, n_4, n_5\}$ with the last location in these partial trajectories as the current location. These partial trajectories form the training dataset. We choose four representative features for each trajectory in the training dataset (Section 3.1) and group trajectories with identical features into clusters (Section 3.2). When a new query trajectory arrives, our algorithm maps it to a cluster according to its features, and predict destination based on the average destination of all training trajectories in that cluster (Section 3.3).

3.1. Feature Selection

We find that the most prominent features of a trajectory that decides the predicted destination using the trajectory is the direction of travel. Specifically, we select the following two pieces of directional information: (i) the direction from the starting location to the current location in the partial trajectory training dataset; and (ii) the direction of travel at the current location (i.e., instantaneous direction). These two directional features are categorised into North, East, South, and West.

Besides the two directions, we select the current location as an essential feature. Together, the first three features provide key insight to the user's destination. For instance, when a user is on the city-airport freeway and the direction is towards the airport, then it is very likely that the destination is the airport.

The last feature we select is the travel distance. We would like to separate short-distance trips from long distance trips because we observe that longer trajectories and shorter trajectories passing through the same locations have very different destinations.

We summarise the four selected features in the list below.

- The instantaneous travel direction at the current node θ_c;
- The direction from the starting node to the current node $\theta_{s \to c}$;
- the location of the current node n_c; and
- the travel distance from the starting node to the current node in ℓ_1 distance $d_{s \to c}$.

3.2. Training

Based on the four aforementioned features, we partition the training dataset into different clusters. The method of clustering is by hashing, where partial trajectories with the same features will be put into the same cluster. In order words, all partial trajectories in a single cluster have the same four features. In each cluster, we need to find a location that will be used as the predicted destination for that cluster. Since we use distance deviation to measure prediction accuracy, it makes sense to use a location $(\hat{x}, \hat{y})$ in each cluster that minimises the average ℓ_1 distance from the destinations of all training trajectories in that cluster.

Let k denote the number of trajectories in a cluster and (x_i, y_i) denote one of the destinations of trajectories in that cluster where $i \in [1, k]$. Then the predicted destination for this cluster is defined as:

$$(\hat{x}, \hat{y}) = \arg\min_{(\hat{x},\hat{y})} \frac{1}{k} \sum_{i=1}^{k} |\hat{x} - x_i| + |\hat{y} - y_i|.$$

Because the expressions of x and y can be separated, finding $(\hat{x}, \hat{y})$ is equivalent to finding an optimal $\hat{x}$ and an optimal $\hat{y}$ separately. It is straightforward to observe that the optimal value of $\hat{x}$ is the median of $x_1, x_2, \ldots, x_k$, and the optimal value for $\hat{y}$ is similar. Therefore, the predicted destination of a cluster is the point that have the median latitude and the median longitude of all destinations in that cluster.

3.3. Prediction

When given a query trajectory, we first compute its feature values and find the cluster it belongs to by matching its four features. Then, we simply return the computed predicted destination of that cluster.

In the case where a query trajectory cannot be categorised to any cluster due to insufficient training data, we use the current node as the predicted destination. The same action is performed for SubSynEA when comparing these two algorithms in the experimental study in Section 6.

3.4. Discussion

The PROFILE method is more task-oriented than the SubSyn method for two reasons:

- PROFILE is a discriminative method that models the destination probability more directly.
- PROFILE treats the destination prediction problem as a regression problem. In contrast, SubSyn converts it to a classification problem that aims to find the cell with the highest probability containing the true destination.

Therefore, it makes sense that PROFILE outperforms SubSyn method in terms of effectiveness, which is consistent with the experimental results presented in Section 6.

4. Algorithm

In this section, we present the training and prediction algorithms in detail.

4.1. The Training Algorithm

The training algorithm takes a dataset $\mathbb{D}$ of historical taxi trajectories and a grid size g as input, and returns a matrix A containing the average destinations of all clusters. The algorithm is presented in Algorithm 1.

Algorithm 1: PROFILE-Training($\mathbb{D}$, g)

```
1  for all cluster do                                   // initialization
2      A[cluster], count[cluster] ← cluster.curCell.center, 0
3  for T in 𝔻 do                          // count trajectories for every cluster
4      for T^p in T do
5          calculate feature values and find the cluster that T^p belongs to
6          count[cluster] ← count[cluster] + 1
7  for all cluster do                     // calculate the position of median
8      count[cluster] = ⌊(count[cluster] + 1)/2⌋
9  back up count matrix
10 sort trajectories in 𝔻 by the latitude of destination
11 for T in 𝔻 do                  // find median latitude for every cluster
12     for T^p in T do
13         calculate feature values and find the cluster that T^p belongs to
14         count[cluster] ← count[cluster] − 1
15         if count[cluster] = 0 then
16             A[cluster].latitude = T.dst.latitude
17 restore count matrix from back up
18 sort trajectories in 𝔻 by the longitude of destination
19 for T in 𝔻 do                 // find median longitude for every cluster
20     for T^p in T do
21         calculate feature values and find the cluster that T^p belongs to
22         count[cluster] ← count[cluster] − 1
23         if count[cluster] = 0 then
24             A[cluster].longitude = T.dst.longitude
   return: A
```

Then we explain the algorithm line by line. In lines 1-2 of Algorithm 1, we set a default average destination and trajectory count for every cluster. A cluster here is identified by four feature values (n_c, $d_{s\to c}$, θ_c, $\theta_{s\to c}$). We set the default average destination of *cluster* ($A[cluster]$) to the center of n_c of *cluster* (*cluster.curCell.center*) and set count[*cluster*] to 0.

In lines 3-6, we traverse all partial trajectories that are extracted from all trajectories in the dataset $\mathbb{D}$. In order to calculate value of features in practice, we only need to extract consecutive GPS points in T instead of the entire T^p. In this way, θ_c and $\theta_{s\to c}$ can be calculated directly.

In lines 7-8, we calculate the position of the median for every cluster. For example, the position of the median of a cluster containing 7 or 8 partial trajectories should be the x and y coordinates of the 4^{th} location. Then, we find the median latitude and longitude of destinations for every cluster based on the position of median we store in the matrix count[$\cdot$]. Because the matrix count[$\cdot$] would be modified in the loop from line 11 to line 16, we first make a back-up for it at line 9 and restore it from back-up on line 17.

In lines 10-16, we first sort all the trajectories in the dataset $\mathbb{D}$ by latitude followed by traversing all T^p (similar to lines 3-6). When a T^p is extracted from T, we decrease the trajectory count of the its cluster by 1. If the trajectory count of a *cluster* comes down to 0, the latitude of destination of the current T^p is the median latitude for that *cluster*. Finally, we store the median latitude in matrix $A[cluster]$.

Lines 19-24 are used to compute the median longitudes. The method is similar to lines 10-16. Eventually, the algorithm returns the matrix A, which stores the predicted destination (median latitude, median longitude) of every cluster.

4.2. The Prediction Algorithm

Compared with the training algorithm, the prediction algorithm is much more straightforward. The prediction algorithm is summarised in Algorithm 2.

Algorithm 2: PROFILE-Prediction(A, T^p)

if T^p *contains only one point* **then**
  return: $T^p.src$
$L \leftarrow 1 + \sum_{i=1}^{c} \text{CELL}\ell_1(p_{i-1}, p_i)$
$\theta_c \leftarrow \angle(p_{c-1} \rightarrow p_c)$
$\theta_{s\rightarrow c} \leftarrow \angle(p_s \rightarrow p_c)$
return: $A[n_c, \max\{L, g\}, \theta_c, \theta_{s\rightarrow c}]$

Given a partial trajectory T^p, the algorithm first calculates the value of the four features. Then, it returns the predicted destination of the cluster that T^p belongs to. One exception is that T^p contains only one point, i.e. the starting location. In this case, the algorithm simply predicts the staring location as the destination for that the information is inadequate.

5. Cost Analysis

In this section, we analyse both the time and space complexities of the two algorithms described in this paper, i.e., PROFILE-Training and PROFILE-Prediction.

In the following analysis, we let $\mathbb{D}$ be the trajectory dataset and s be the average number of GPS points in each trajectory.

5.1. The Training Algorithm

Time Complexity: As we described in Section 4, Algorithm 1 sorts all trajectories in $\mathbb{D}$ twice (by the latitude of destinations and the longitude of destinations, respectively) and traverses all T^p in $\mathbb{D}$ three times. Because the latitude and longitude in the dataset we use have at most six digits after the decimal point (otherwise it is also reasonable to truncate them to such precision which is enough for daily life), we treat them as integer after multiplying them by 10^6. Then we use a 2-pass sorting mechanism called the *Pigeonhole Sort* [11], whose time and space complexities are both $O(C + |\mathbb{D}|)$ where C is the range of either latitude or longitude and thus a constant. Therefore, the sorting process runs in $O(2|\mathbb{D}|)$. Regarding the traversing process, the algorithm needs to process every GPS point in the trajectory dataset. Thus, traversing all the dataset runs in $s|\mathbb{D}|$ where s is the average number of GPS points in a trajectory. Therefore, the total time complexity is $O(|\mathbb{D}| \times 2 + s|\mathbb{D}| \times 3) = O(s|\mathbb{D}|)$.

Space Complexity: In Algorithm 1, the matrices we used include A, count$[\cdot]$ and a back-up for count$[\cdot]$. They all have the same size of $g^2 \times g \times 4 \times 4$ but different types in that A is a pair of `float` numbers (latitude and longitude) whereas count$[\cdot]$ and its back-up are `int`. Therefore, the total space needed is $16g^3 \times 2 \times 4 + 2 \times 16g^3 \times 4 = 256g^3 = 256m^{1.5}$ Bytes, which is in $O(m^{1.5})$.

5.2. The Prediction Algorithm

Time Complexity: Algorithm 2 first calculates the value of the selected features for a partial trajectory T^p, i.e. n_c, $d_{s\rightarrow c}$, θ_c and $\theta_{s\rightarrow c}$, which can be calculated in constant time. Then the algorithm returns the corresponding destination stored in A. The whole process is as easy as looking up an element in a table. Therefore the time complexity of Algorithm 2 is $O(1)$.

Space Complexity: Algorithm 2 only requires matrix A to be stored in the memory. The size of A is $g^2 \times g \times 4 \times 4 = 16g^3 = 16m^{1.5}$, and every element in A is a pair of *float* numbers (latitude and longitude). Therefore, the space required is $16m^{1.5} \times 2 \times 4 = 128m^{1.5}$ Bytes and the space complexity is $O(m^{1.5})$.

5.3. Summary

We summarise the time and space complexities in the tables below. We also compare the complexities of our algorithms with those of the SubSyn methods. Table 2 summaries the time and space complexity of the algorithms, and Table 3 provides an intuitive display of the space size that the algorithms may consume on grids of different sizes. From these tables we can see that PROFILE-Prediction outperforms SubSynEA-Prediction in terms of both runtime efficiency and space occupation. PROFILE-Training also outperforms SubSynE-Training in term of space complexity. The comparison on runtime efficiency is less intuitive, and we will show the experimental results in Section 6.

Complexity	Time	Space
SubSynE-Training	$O(m^{2.5})$	$O(m^2)$
PROFILE-Training	$O(s\|\mathbb{D}\|)$	$O(m^{1.5})$
SubSynEA-Prediction	$O(m)$	$O(m^2)$
PROFILE-Prediction	$O(1)$	$O(m^{1.5})$

Table 2. Time and Space Complexities

Grid Size	40	50	60	70
SubSynE-Training	78M	191M	396M	733M
PROFILE-Training	16M	31M	53M	84M
SubSyn-Prediction	39M	95M	198M	366M
SubSynEA-Prediction	332M	810M	1.6G	3.0G
PROFILE-Prediction	8M	15M	26M	42M

Table 3. Space Occupation for Various Grid Sizes

6. Experimental Study

In this section, we evaluate both the runtime efficiency and prediction accuracy of SubSynEA and PROFILE. In Section 6.1, we present the dataset and means of measurement used in experiments. Then, we compare the runtime efficiency of both the training and prediction algorithms in Section 6.2 and Section 6.3, respectively. Finally, we compare the prediction accuracy of the two algorithms in Section 6.4.

6.1. Setup

Dataset: The dataset we use is a real-life taxi GPS trajectory dataset containing nearly $900,000$ trajectories from the *T-drive* project [12, 13]. The taxi trajectory dataset is from the city of Beijing within a $40km \times 40km$ area. We randomly select $20,000$ trajectories from the training dataset as the test dataset.

Parameters: We conduct experiments by varying the grid size g from 20-70 to study the effect of different resolutions. When $g = 20$, the resolution of each cell is $2km$. When $g = 70$, the resolution becomes much finer to approximately $570m$. The runtime efficiency is simply a measure of running time. The algorithms were run on a workstation machine with Intel™ Xeon-W3670 CPU (3.2GHz) and 24GB RAM. The prediction accuracy is measured by the distance deviation between the predicted destination and the true destination of all query trajectory. The deviation is measured in ℓ_1 distance because of the grid representation.

6.2. Efficiency of Training Algorithms

Fig. 2 shows the running time of the two training algorithms, PROFILE and SubSynEA. It can be observed that when g is small (e.g., $20 - 40$), the difference between the running times of SubSynEA-Training and PROFILE-Training is within 10 seconds. As g increases, the difference becomes more significant (e.g., when $g = 70$, PROFILE-Training outperforms SubSynEA-Training by more than an order of magnitude). It is worth mentioning that varying the grid size only has marginal influence on the running time of PROFILE-Training since its training time is mostly associated with the size of the training dataset rather than the grid size.

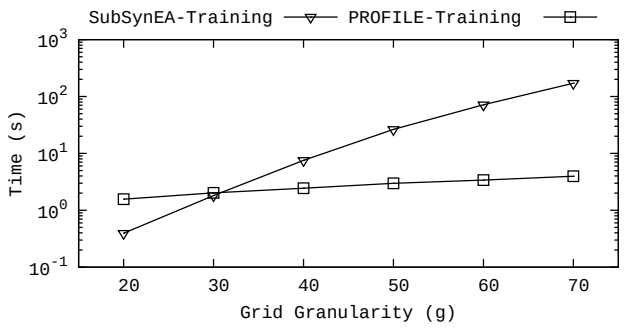

Figure 2. Runtime Efficiency of the Training Algorithms

6.3. Efficiency of Prediction Algorithms

We also compare the runtime efficiency of SubSynEA-Prediction and PROFILE-Prediction in the average time to predict the destination for one query in Fig. 3. By using the already-determined predicted destination from the training stage, PROFILE-Prediction runs in constant time, whereas SubSynEA-Prediction needs to compute the probability of every cell containing the true destination. Therefore, PROFILE-Prediction can run one to two orders of magnitude faster constantly.

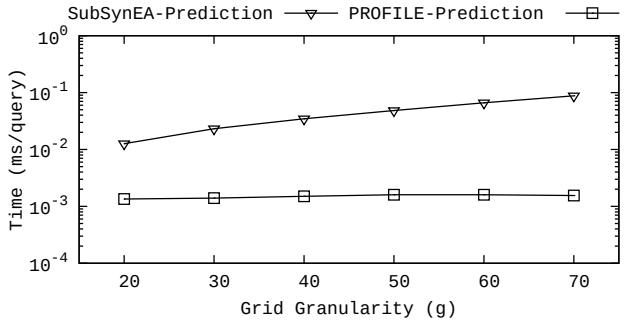

Figure 3. Runtime Efficiency of Prediction Algorithms

6.4. Accuracy

In this part, we compare the prediction accuracy of the PROFILE method with that of SubSynEA. The comparison is conducted by varying the grid size g from 20 to 70. Furthermore, since the length of trajectories matters as discussed in Section 3, we also vary the length of trip completed percentage. The distance deviation of PROFILE-Prediction is about $3km$ smaller than that of SubSynEA-Prediction when the prediction is made at 30% of the trip and $1km$ smaller at 70% of the trip.

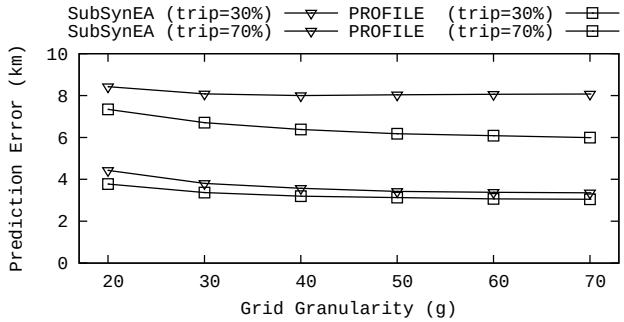

Figure 4. Prediction Accuracy

As shown in Fig. 4, PROFILE-Prediction outperforms SubSynEA-Prediction constantly (about $3km$ when trip% = 30% and $1km$ when trip% = 70%).

7. Conclusion

In this paper, we proposed a novel discriminative method called *PROFILE*, which first selects four features for the

training trajectory dataset and partitions the dataset into clusters based on these features. Then, we compute a predicted destination for each cluster using the median of all destinations in that cluster. When given a query trajectory, the PROFILE method find the cluster that has the identical features as the query trajectory, and return the predicted destination of that cluster to the user.

Experiments based on real-life taxi GPS trajectory dataset has shown that the performances of our PROFILE method is better than that of the SubSynEA method in terms of both runtime efficiency and prediction accuracy. Furthermore, the time complexity of the PROFILE-Training algorithm is linear to the size of training set and the PROFILE-Prediction algorithm runs in constant time.

References

[1] XUE, A.Y., QI, J., XIE, X., ZHANG, R., HUANG, J. and LI, Y. (2014) Solving the data sparsity problem in destination prediction. *The VLDB Journal* .

[2] NG, A.Y. and JORDAN, M.I. (2001) On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*: 841–848.

[3] KRUMM, J. and HORVITZ, E. (2006) Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Ubiquitous Computing* (Springer), 243–260.

[4] KRUMM, J., GRUEN, R. and DELLING, D. (2013) From destination prediction to route prediction. *Journal of Location Based Services* **7**(2): 98–120.

[5] ZIEBART, B.D., MAAS, A.L., DEY, A.K. and BAGNELL, J.A. (2008) Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing* (ACM): 322–331.

[6] 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 *29th International Conference on Data Engineering (ICDE)* (IEEE): 254–265.

[7] XUE, A.Y., ZHANG, R., ZHENG, Y., XIE, X., YU, J. and TANG, Y. (2013) Desteller: A system for destination prediction based on trajectories with privacy protection. *Proceedings of the VLDB Endowment* **6**(12): 1198–1201.

[8] ALI, M.E., ZHANG, R., TANIN, E. and KULIK, L. (2008) A motion-aware approach to continuous retrieval of 3d objects. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on* (IEEE): 843–852.

[9] ZHANG, J., TAO, X. and WANG, H. (2014) Outlier detection from large distributed databases. *World Wide Web* **17**(4): 539–568.

[10] YAO, W., HE, J., WANG, H., ZHANG, Y. and CAO, J. (2015) Collaborative topic ranking: Leveraging item meta-data for sparsity reduction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[11] BLACK, P.E. (2006), Pigeonhole Sort, `http://www.nist.gov/dads/HTML/pigeonholeSort.html`. In Dictionary of Algorithms and Data Structures [online].

[12] YUAN, J., ZHENG, Y., ZHANG, C., XIE, W., XIE, X., SUN, G. and HUANG, Y. (2010) T-drive: Driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10: 99–108.

[13] YUAN, J., ZHENG, Y., XIE, X. and SUN, G. (2011) Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11: 316–324.