EXAMENSARBETE INOM THE BUILT ENVIRONMENT,
AVANCERAD NIVÅ, 30 HP

STOCKHOLM, SVERIGE 2018

Travel Diary Semantics Enrichment of Trajectories based on Trajectory Similarity Measures

RUI LIU

KTH
SCHOOL OF ARCHITECTURE AND THE BUILT ENVIRONMENT

TRITA nr SoM EX 2017-47

ROYAL INSTITUTE
OF TECHNOLOGY

Travel Diary Semantics Enrichment of Trajectories based on Trajectory Similarity Measures

Master Degree Thesis

RUI LIU

Division of Geoinformatics
Department of Urban Planning and Environment
School of Architecture and the Built Environment
KTH Royal Institute of Technology
ruiliu@kth.se

Stockholm 2017

Abstract

Trajectory data is playing an increasingly important role in our daily lives, as well as in commercial applications and scientific research. With the rapid development and popularity of GPS, people can locate themselves in real time. Therefore, the users' behavior information can be collected by analyzing their GPS trajectory data, so as to predict their new trajectories' destinations, ways of travelling and even the transportation mode they use, which forms a complete personal travel diary. The task in this thesis is to implement travel diary semantics enrichment of user's trajectories based on the historical labeled data of the user and trajectory similarity measures. Specially, this dissertation studies the following tasks: Firstly, ***trip segmentation*** concerns detecting the trips from trajectory which is an unbounded sequence of timestamp locations of the user. This means that it is important to detect the stops, moves and trips of the user between two consecutive stops. In this thesis, a heuristic rule is used to identify the stops. Secondly, ***tripleg segmentation*** concerns identifying the location / time instances between two triplegs where / when a user changes between transport modes in the user's trajectory, also called makes transport mode transitions. Finally, ***mode inference*** concerns identifying travel mode for each tripleg. Specially, steps 2 and 3 are both based on the same trajectory similarity measure and project the information from the matched similar trip trajectory onto the unlabeled trip trajectory. The empirical evaluation of these three tasks is based on real word data set (contains 4240 trips and 5451 triplegs with 14 travel modes for 206 users using one week study period) and the experiment performance (including trends, coverage and accuracy) are evaluated and accuracy is around 25% for trip segmentation; accuracy varies between 50% and 55% for tripleg segmentation; for mode inference, it is between 55% and 60%. Moreover, accuracy is higher for longer trips than shorter trips, probably because people have more mode choices in short distance trips (like moped, bus and car), which makes the measure more confused and the accuracy can be increased by nearly 10% with the help of reverse trip identifiable, because it makes a trip have more similar historical trips and increases the probability that a new unlabeled trip can be matched based on its historical trips.

Keyword: trajectory similarity measures; trip segmentation; tripleg segmentation; mode inference.

Acknowledgments

I would like to thank Associate Professor Gyözö Gidofalvi and Professor Yifang Ban for their valuable and constructive suggestions. I would also like to thank my friends, Qin Zhang, Chenyang Wang and Yuxin Li. They have given me great support for my thesis.

Content

List of Figures

List of Tables

1 Introduction

1.1 Background

Trajectory data in the current big data age plays an increasingly important role. Travel survey has always been a very important tool of collecting trajectory data to investigate the travel needs and behaviors of the people. However, traditional travel survey methods, like asking people to fill in a standard travel diaries questionnaire, 1) are error prone (inaccurate geocoding, inaccurate time estimates, forgotten trips), 2) are costly, 3) have taken too much time and effort which makes them unsuitable to perform surveys for extended periods and results in a drastic decrease in response rates (Prelipcean 2015).

However, with the rapid development of Global Positioning System (GPS) technology, the current location information of the user can be easily obtained, such as the latitude and longitude, time, speed and direction of the current location of the user. Moreover, with the use of smart phones more usual than before, you can easily use smart phones to collect GPS data with the embedded GPS technology in them. User's route in a certain period of time can be numerous GPS data points, the collection of these GPS data points is the user's trajectory in a certain period of time, each user's mobile trajectories can be saved through the smart phone. This kind of motion process is usually recorded as a series of timestamp points (*x, y, t*), where *x* and *y* is the coordinate information, which together indicate a two-dimensional point, and the *t* is the real time of the point (Gong 2015), showed in Figure 1.1. A variety of time-series trajectory data also spawn a lot of interesting applications, such as finding potential friends through the daily life trajectory similarity (Quannan 2008), through the human body with the sensor to collect the data for action identification (Allen 2009), and climate change prediction (Mohammad 2014), etc.

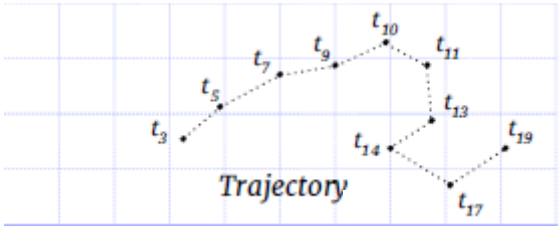

Figure 1.1: Example of the trajectory data, the location information together with the real time t_i of the point (Source Kucuk 2016)

Because of the development of Internet technologies and mobile communication, positioning, and sensing, information (like locations and accelerometer readings) can be collected easier and more accurately at a higher level of (spatial and temporal)

detail than before. Then different inference methods can turn the collected information into travel diaries which contains information about travel relevant entities (trips, destinations, triplegs, travel modes, etc.). These inference methods often 1) segment the measurement sequence based on object stationarity criteria or measurement gaps (GPS signal loss indoors), 2) segment trips into triplegs based on abrupt changes in the measurements time-series, 3) infer mode of either a measurement or the measurements of a trip leg using machine learning or heuristic methods on features derived from the measurement(s), and 4) infer the destination of the trip as a previously visited place or Point Of Interest (POI) that is closest to the last measured location of the trip (Prelipcean 2016a).

With the development of the network, many mobile phone users are increasingly willing to share their own trajectory with others (Ying 2010). When a user uploads his/her trajectory diary, these referral systems can analyze his/her friends' information to recommend similar user activity to him/her. Moreover, these recommendation systems not only recommend the behavior of this user's friends, but also other similar user's behavior. It is shown in Figure 1.2 that a user is uploading his trajectory and the system can recommend some other users to him with the similar trajectory. It is clear that the similarity between user trajectories plays a very important role in these recommendation systems.

Figure 1.2: A trajectory uploaded by a user

1.2 Objectives

The work in this thesis is based on the basic idea of the proposed inference

approach ,which is that a trip that has a (geographically and temporally) very similar route to a previously historical mode-segmented/labeled trip will likely have the same destination as well as transport mode sequence / segmentation. It is all known that human has habits; they often get used to do something and do it in the exactly same way. Quite often user's trips are habitual. For example, when people go to work or school from home, if they take public transport, they often follow the same route and the same modes even the same time; if they drive their own car, they also often drive on the same road even if there are other options. As a result, if the dataset of a user's travel diary is large enough, when the user has a new trip trajectory, the information of the trajectory (like destination, tripleg segmentation, mode choice, etc.) can be inferred from previous similar trip trajectories of the user.

The aim of this project is to explore how effectively one can use the similarity of historical labeled trajectories to enrich trajectories with travel diary semantics (tripleg and travel modes). The specific goals are to 1) adopt suitable trajectory similarity measures for semantic enrichment; 2) provide a prototype implementation of the trajectory similarity based semantic enrichment in a spatial database; 3) evaluate the accuracy of the trajectory similarity based semantic enrichment on real word datasets under different configurations.

1.3 Thesis structure

This thesis is structured as follows. Chapter 1 introduces the thesis and research objectives. Chapter 2 provides a detailed review of the literature about the similarity measures and the methods of travel diary semantics enrichment of trajectories. Chapter 3 illustrates the methodology. Chapter 4 presents the three main implementations and analyzes the results. Chapter 5 makes the conclusion of the thesis. Finally, Chapter 6 presents future research directions in this field of study.

2 Related Work

2.1 Trajectory similarity measures

In a general sense, similarity is a measure of the degree of similarity between two objects, generally expressed by distance. As a result, the distance between trajectories can also be regarded as a representation of the similarity between trajectories. There are many research results in terms of trajectory similarity calculation.

Sum-of-Pairs Distance. As early as 1993, Agrawal et al. proposed a representation of the similarity between trajectories based on the Euclidean distance. The method requires that the sampling points of the two trajectories are one-to-one, which means

that both the sampling interval and the number of sampling points (i.e., the trajectory length) is the same. The distance between the trajectories is obtained by summing up the distances between the corresponding pairs of points, showed as Equation 2.1.

$$SPD\ (A, B) = \sum_{i=0}^{n} d(a_i, b_i) \tag{2.1}$$

Where A and B are two trajectories with the same number of points; $d(,)$ is the Euclidean distance between two corresponding points (Deng 2011). This method is very simple and easy to implement, but it is relatively sensitive to outliers, since all points including noises are required to match. Moreover, the two trajectories must be the same length (which means equal number of points contained in the trajectory) and it does not implement the local time shifting.

Dynamic Time Warping Distance. In order to overcome the drawback of the requirement of two trajectories to be the same length, Dynamic time warping (DTW) distance is proposed. The basic idea of DTW is to allow "repeating" some points as many times as needed in order to get the best alignment (Deng 2011) and so it does not need the two trajectories to be the same number of points. For example, in the first two cases of the third (recursive) case in Formula 2.2, the recursive call leaves on the trajectories unchanged, in other words, their heads will be matched or will be repeated multiple times.

The DTW distance is defined as Equation 2.2:

$$DTW(A, B) = \begin{cases} 0, \text{ if } n=0 \text{ and } m=0 \\ \infty, \text{ if } n=0 \text{ or } m=0 \\ d\big(Head(A), Head(B)\big) + \min \begin{cases} DTW\big(A, Rest(B)\big) \\ DTW(Rest(A), B) \\ DTW(Rest(A), Rest(B)) \end{cases} \end{cases} \tag{2.2}$$

Where A and B are two trajectories with length of n and m, $Head\ (A)$ is the first point of trajectory A, $Rest\ (A)$ is the rest point of trajectory A, the same with the trajectory B, $d()$ is the distance function between two points.

This method does not require the two trajectories to be the same length. However, it is still sensitive to noise (outliers) just like Sum-of-Pairs Distance.

Similarity Algorithms Based on Editing Distance. Editing distance is a concept derived from text processing. It refers to the minimum operations required to change a text sequence by adding, deleting, and changing into another sequence (Crochemore 1994). Chen (2004, 2005) have made improvements on the basis of editing distance by removing the noise effects by quantizing the distance between a pair of trajectory point to two values, 0 and 1, and proposed the ERP (Edit distance with Real Penalty) and EDR (Edit Distance on Real Sequence) distance measures. Another kind of editing distance is the longest common sub-sequence distance (LCSS), whose basic idea is to allow skipping over some points rather than just rearranging them (Deng

2011).

Methods based on editing distance does not require the same length of two trajectories and can also reduce the effect of outliers, especially for EDR distance, but these methods' results are generally less intuitive and difficult to be interpreted.

2.2 Trip detection

According to Parent et al. (2013), trajectory segmentation depends on the application of the services for which it is performed. In general, the movement in a trajectory can be divided into two periods, one is that the object is stationary, the other one is that the object is moving, which is known as stops and moves (Prelipcean 2016b). With these stops and moves, the start and end point of a trip can be known and a trip can be detected.

A heuristic rule to distinguish the stops and moves is called the stationary rule, i.e., if the speed is very low (the thresholds of the "low" can be different) for longer than a period of time, then the end of a trip has been identified (Prelipcean 2016a). Regarding the choice of the time threshold for trip detection, researchers have used different values, like 300 seconds, 900 seconds and 120 seconds which are mostly common used by Wolf (2000), Tsui and Shalaby (2006), Stopher (2008) and Rasmussen (2015). In this thesis, this stationary rule is also used to identify the trips.

2.3 Tripleg detection and mode inference

For tripleg detection and mode inference, there are two main approaches: segment based inference and point based inference. For segment based inference, a trip is split into potential triplegs and then chooses the travel mode for these triplegs with some further methods. For point based inference, each point of the trajectory is segmented into a transportation mode; as a result, the tripleg detection can be obtained at the same time, which can be identified by the maximal sequence of consecutive points with the same transportation mode (Prelipcean 2016a).

2.3.1 Segment based inference

For segment based inference, in general, it has broken down the task into three steps: segmenting trajectories into trips (which has been mentioned in the previous section), splitting trips into triplegs, and classifying triplegs. There are also a wide variety of approaches for each of three tasks.

For tripleg segmentation in segment based inference, the key is to find the mode change point. Chung and Shalaby (2005) first define how to find the mode change

point (known as transfer point): 1) find the first recorded location with speed faster than 10 km/h and time difference with the previous location is larger than 5 seconds, or 2) find a blockage (which means no locations recorded during this period, except the end and the start point), and then set the first point as a change point if the blockage distance is more than 150 meters and the speed is more than 10 km/h. After Chung and Shalaby (2005), there are also some scientists proposing some other heuristics. However, most of them are similar to Chung's. Until Zheng et al. (2010), these authors proposed that people have to walk when they make the transfer. So firstly, they segment triplegs into walk and non-walk triplegs, and then they set upper bounds for velocity and acceleration of the walk triplegs (2.5 m/s for speed, and 1.5 m/s2 for acceleration) to identify the tripleg.

For classifying triplegs (mode detection), there are different methods to implement it, like decision trees, fuzzy logic, random forests, rule-based classifiers and membership functions. However, the reported precision values vary with the methods and the number of modes choices, like 94% using fuzzy logic for four modes (Tsui and Shalaby 2006) and 91.6% using membership functions for ten modes (Biljecki et al. 2013).

2.3.2 Point based inference

Point based inference has been widely used in the field of Location Based Services. As mentioned before, unlike the segment based inference, point based inference segments every point of the trajectory into modes, it does not generate the triplegs. However, the triplegs can be identified as the maximal sequence of consecutive points with the same transportation mode.

Concerning to the mode detection, the methods used in segment based inference can also be used, like random forests (Stenneth and Xu 2011) with 92.8% accuracy and 92.9% recall for six transportation modes, decision tree (Wang et al., 2010) with 70% precision for six transportation modes, AdaBoost together with Decision Tree (Hemminki et al., 2013) with 80.1% accuracy and 82.1% recall for seven transportation modes.

In this thesis, the idea of the segment based inference will be used, where the task is divided into three steps: segmenting trajectories into trips (which has been mentioned in the previous section), splitting trips into triplegs, and classifying triplegs. The method proposed in this article is based on the trajectory similarity measures and historical labeled trips.

3 Methodology

The overview of the methodology is showed in Figure 3.2. Firstly, based on the real world dataset (information about location, time and speed of the user), basic data processing is implemented to form a trajectory with these collected data. Secondly, trip segmentation is done based on a heuristic rule. Thirdly, the trajectory similarity measure is defined. Lastly, based on similarity measure, tripleg segmentation and mode inference are implemented. The detailed descriptions of each step are discussed in the following sections.

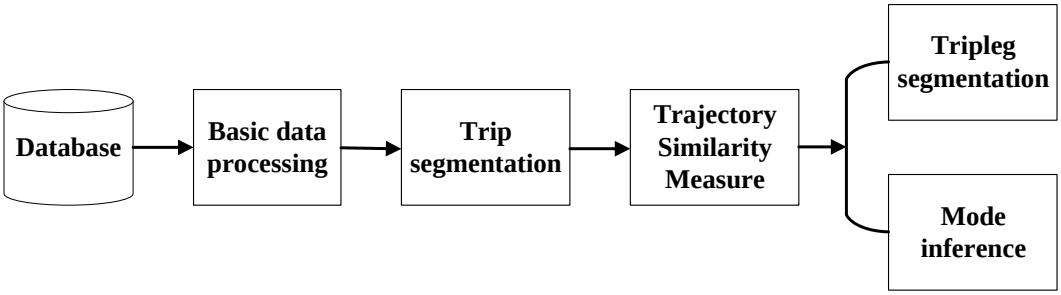

Figure 3.1: Overview of the proposed methodology

3.1 Basic data processing

The original information about each point is (latitude, longitude, time), which can be represented as (*x, y, t*). With the function st_makepoint() in PostgreSQL, the point's location can be changed into geometry type. Together with the time, the location point can be expressed as timestamp-geometry pairs, called "tg_pair" (g as the geometry, t as the time). Given the start point and the end point of a trip or tripleg, the collection of tg_pairs between start and end point forms a data type called "Trajectory" (this process is done with an "ordered" collection in order to guarantee the order between the points of a trajectory) and this new data type stores the time dimension of a trajectory as a 3rd dimension of a 3d polyline. The data model of two data types is showed in Figure 3.1. The algorithm used to form a trajectory from a collection of points can be seen in Appendix Algorithm 1.

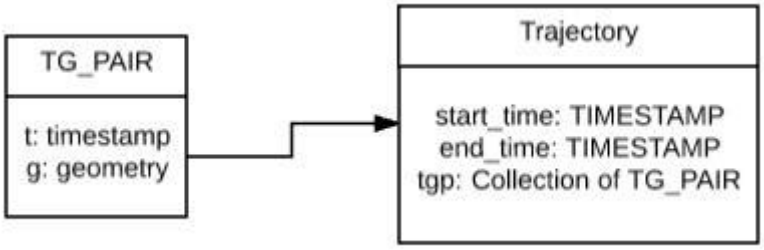

Figure 3.2: Data model of two abstract data types - TG_PAIR and Trajectory (Kucuk et al. 2016)

After obtaining the "Trajectory" data type of each trip and tripleg, the number of the locations contained in the trajectory and length of the trajectory are calculated using Algorithm 2 and 3 showed in Appendix.

3.2 Stationarity criteria based trip segmentation

Trip segmentation is based on the heuristic rule, called stationary rule, if the speed is very low (smaller than 3.6km/h, which has been used by Prelipcean (2016a)) for longer than a period of time (120s, which has been used by Wolf (2000), Tsui and Shalaby (2006), Stopher (2008) and Rasmussen (2015)), then the end of a trip is identified.

For example, as mentioned before, trajectory data can been seen as the collection of the points (*x, y, t*) which where *x* and *y* is the coordinate information and the *t* is the real time of the point. As a result, a trajectory can be plotted in a 3-D dimension space intuitively. Figure 3.2 is a part of a user's trajectory.

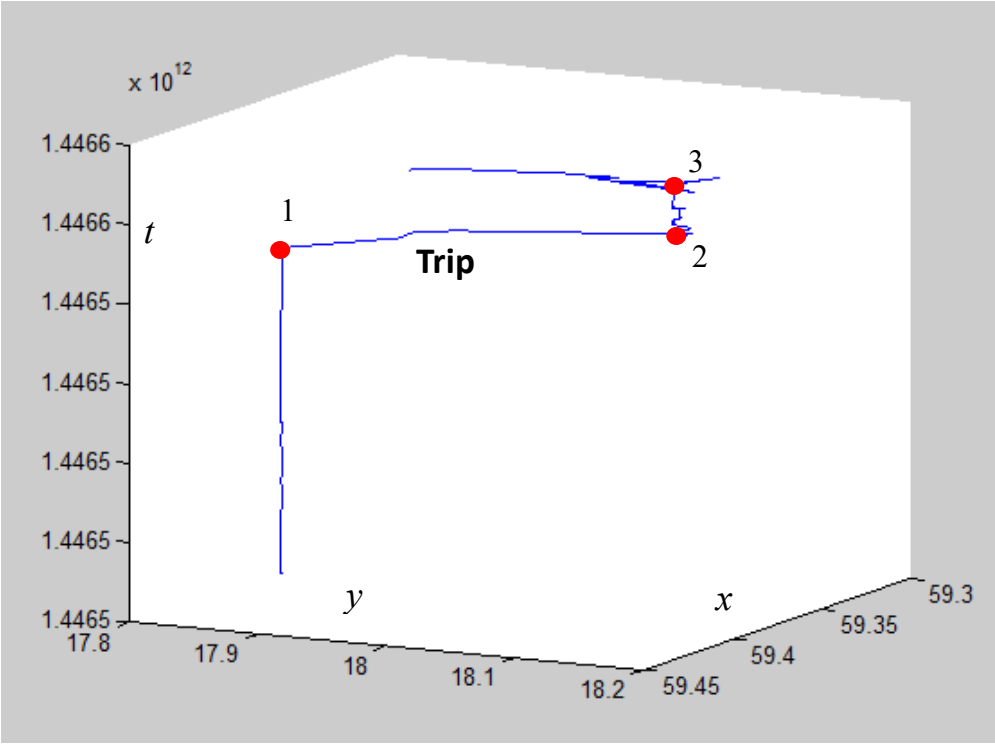

Figure 3.3: Part of a user's trajectory plotted in Matlab. x and y are coordinates of the user's locations at different timestamps t measured in milliseconds

From the Figure 3.2, a trip can be detected intuitively, that is the part of trajectory between point 1 and point 2. Because as it can be seen in the figure, there is a very long dwelling time of the user before point 1 (definitely larger than 120s), in this period of time, the user barely moves. So the point 1 can be set as a start point of the trip. Then there is also a long nearly no-moving period of dwelling time between point 2 and point 3, so location near point 2 can be identified as an end point. With the

start point 1 and end point 2, a trip is now detected. This approach is very intuitive and easy to understand, however, it is hard to know the exactly location of point 1 or point 2. Moreover, it is also very hard to find some small distance trips. However, a database implementation can be fast, simple and elegant and can be executed in the same system where the data is collected and stored.

In a database implementation, for trip segmentation, the first step is to classify the state of each point into stop or moving. As mentioned before, if a user's speed is less than 3.6 km/h for more than 2 minutes, then that represents a dwell period, which indicates the stop statement, otherwise, it is the moving statement. The algorithm used to implement this is in Appendix Algorithm 4. After this, it is necessary to classify these stop point into start point and end point so that a trip can be detected. For a point, if the latter consecutive 5 points (nearly 2 minutes time interval) of it are all moving point, then this point should be a start point; if the previous 5 points of it are all moving point, then this point should be an end point; if neither of these two conditions satisfies, the point is an on-trip-point. The Algorithm 5 and 6 is for this idea showed in Appendix. With the identification of the start point and the end point, a trip can be detected. Segmentation information is recorded in a simple but less efficient fashion by annotating each measurement with *stop_point*, *start_point*, and *end_point* Boolean attributes. Admittedly, a suboptimal choice has been made and the adaptation of the proposed methods to a more efficient representation of periods in particular is left for future work.

After trip segmentation, it is important to evaluate the performance and the approach of distance check will be applied between the labeled trip's start / end point and the trip's start / end point identified by the heuristic rule. Features used are shown in Table 3.1.

Table 3.1: List of features used by trip segmentation performance evaluation

Trip_id	Id of the identified trip
user_id	id of the user
id_start_point1	id of the labeled start point of the trip
id_start_point2	id of the identified start point of the trip
id_end_point1	id of the labeled end point of the trip
id_end_point2	id of the identified end point of the trip
disCheck_start	a Boolean indicator $dist(id\_start\_point\ 1\ and\ 2) < \Delta d$
disCheck_end	a Boolean indicator $dist(id\_end\_point\ 1\ and\ 2) < \Delta d$
trip_seg_eva	a Boolean indicator: trip segmentation successfully or not

Specially, *disCheck* is a Boolean indicator (1 if the distance function between two points satisfies, 0 if not) . *Trip_seg_eva* is a Boolean indicator (1 if *disCheck_start* and *disCheck_end* both equal to 1, 0 if else). Then it can be obtained that how many trips can be detected successfully. About this distance function, there are many methods to check whether the predicted segments are matched to the truth segment

for computing the error (Δd). The most commonly method is to use a spatial or temporal buffer around the edges of the ground truth intervals and considers any inferred interval whose both edges fall within the buffer area as a match (Prelipcean 2016a). Figure 3.3 illustrates this idea.

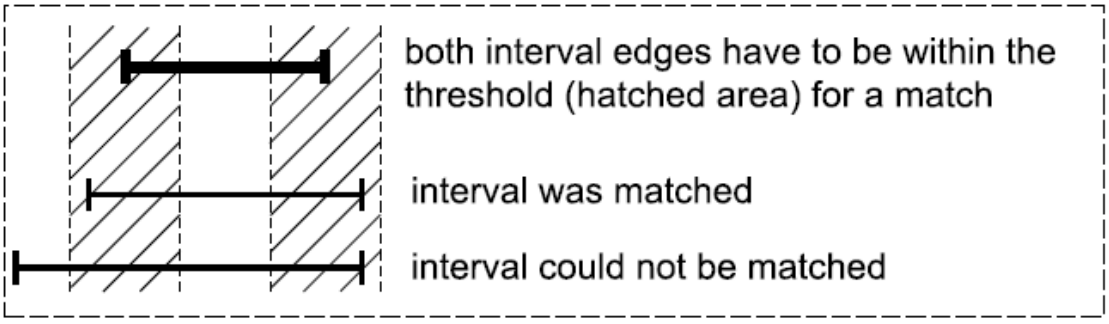

Figure 3.4: The standard of the predicted segments are matched to the truth segment for computing the error (the figure is selected from (Prelipcean 2016a))

Concerns to the accuracy of the trip segmentation, if the distance between the detected start point and the labeled start point is smaller than a threshold, at the same time, the distance between the detected end point and the labeled end point is also smaller than the threshold, then the trip has been detected successfully and set the value of *trip_seg_eva* in Table 3.2 as 1, otherwise as 0. The threshold here is defined as (length of the trip) / 5. Then the accuracy of the trip segmentation is obtained by the percentage of the trips whose *trip_seg_eva* is 1. The Algorithm 7 in Appendix shows the processing.

3.3 Trajectory similarity measure

As mentioned, there are many research results in terms of trajectory similarity calculation, like Euclidean distance, Dynamic Time Warping Distance and different kinds of edit distance. On the one hand, it is impossible that people have the all same length trajectory and the local time shifting is needed; on the other hand, the dataset of the thesis is extremely large and it is more probable that a larger dataset contains an outlier than a smaller dataset, but this is merely due to chance. As a result, EDR (Edit Distance on Real Sequence) is chosen. EDR is relatively insensitive to outliers because the matching threshold reduces the increments to values of 0 and 1 only. Therefore, even if outliers still be processed, each outlier can potentially only increase the EDR value by 1 and not some arbitrarily large values as in Euclidean distance (Toohey and Duckham 2015). It is defined as Equation 3.1:

$$EDR(A,B)=\begin{cases} n, \text{ if } m=0 \\ m, \text{ if } n=0 \\ \min\begin{cases} EDR(Rest(A),Rest(B))+subcost, \\ EDR(Rest(A),B)+1, \\ EDR(B,Rest(A))+1 \end{cases} & \text{otherwise} \end{cases} \quad (3.1)$$

where

$$subcost = \begin{cases} 0, & \text{if } d(Head(A), Head(B)) \le \varepsilon \\ 1, & \text{otherwise} \end{cases}$$

A and B are two trajectories with length n and m (number of the points contained in the trajectory), ε is the matching threshold, $Head(A / B)$ is first point of the trajectory A / B, $Rest(A / B)$ represents trajectory A / B with its first point removed (the trajectory now starts from the second point if one exists, otherwise it now has length 0). The $EDR(A, B)$ is the number of insert, delete, or replace operations that are needed to change A into B (Deng 2011). The $d()$ is calculated between two 3-dimensional points. As a result, similar trajectories will have lower value of EDR. For example, in Figure 3.5, the two trajectories have different number of contained points, but they also can be measured by EDR and edit distance highlights that the four middle points have no match (insertions or deletions) and therefore the two trajectories are not similar based on EDR measure. The key part of the similarity measure algorithm is showed in Appendix Algorithm 8.

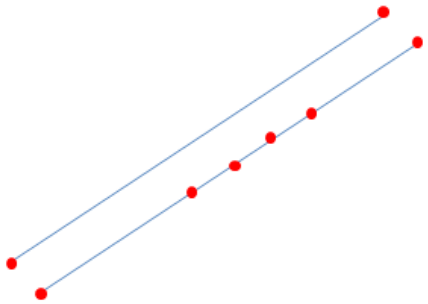

Figure 3.5: One pair of example trajectories.

However, there are several problems that need to be addressed:

Reverse trip. For example, a user's "home-to-work" trajectory and "work-to-home" trajectory are reverse trip to each other. They have the different destinations but they most likely have similar trajectories in 2D Euclidean space (discarding direction). However, due to the 3-dimensional treatment of the distance, the direction and speed of movement is accounted for. In particular, objects that move in opposite direction or at different speeds between two locations will have radically different 3-dimensional trajectories. This means that in the edit distance similarity measures, it can also compare the pair of points with reverse order after time shifting in 3D space. For example, set the first point of trajectory A and the last point of trajectory B as the first pair of point to be compared by EDR measure) then if these two trajectories are reverse trip, they can also have low value of EDR.

Partial trip trajectories connect. For a trajectory A, if there is no candidate similar trip lower than a matching threshold, do not regard it as failing at once, it may find the best candidate trajectory B that provide a good match for first part of trajectory A and find a best candidate C that provides a good match for the rest of trajectory A in

reversed matching order; if the value of EDR measure to *B* and *C* are both lower than a threshold and distance between the last matched point of B and the last matched point of *C* is smaller than a threshold *D,* then regard the trajectory formed by *A* connecting with *B* as the candidate similar trip.

Different meaning of stop segments. The temporal length of "stop segments" might be different meaning. Especially for bus, low movement dynamics might represent either stops at stations or stops in traffic. So the regular stops at similar station locations between two trajectories should increase the similarity of bus mode, the irregular stops in traffic (traffic light, traffic jam) should not decrease the similarity by too much, which means that the matching threshold ε should be larger for mode of bus in EDR similarity measure.

3.4 Trajectory similarity based tripleg segmentation

After defining the similarity measures method and trip segmentation, the tripleg segmentation and mode inference can be carried out. For tripleg segmentation and mode inference, based on the time sequence of each trip, each trip / tripleg of a given user is matched against all the historical trips / triplegs of the user in order and the all results of each trip / tripleg matching (means each trip or tripleg can or cannot be matched by a historical trip or tripleg) are regarded as a population. The performance evaluation is based on this population.

For tripleg segmentation, assume each trip in the data set as a new unlabeled user's trip, then compare it with all his / her historical trips based on similarity measure and return the most similar trip's id and the value of edit distance between them. Then just apply the way of the tripleg segmentation of the most similar historical to the new user's trip. The Algorithm 9 in Appendix shows this procedure. Evaluating the tripleg segmentation performance also uses the idea of distance check introduced in Section 3.1. Features used are shown in Table 3.2.

Table 3.2: List of features used by tripleg segmentation performance evaluation

Trip_id1	Id of the identified trip
trip_id2	id of the most similar trip
num_tripleg1	real number of the triplegs of the identified trip
num_tripleg2	number of the triplegs of the most similar trip
user_id	id of the user
ed	the edit distance of two trips
length	length of the trip expressed in number of locations contained in the trip
id_seg_point1[]	real collect of the segmenting point of the identified trip
id_seg_point2[]	collect of the segmenting point of the similar trip
disCheck	$\sum dist(each\ pair\ of\ segmentation\ point) < \Delta d$

Specially, *disCheck* is the check of the sum of the distance of each pair of corresponding segmenting point, for example in Figure 3.6, two dots are the real segmenting points and two triangles are the segmenting points based on the similarity measures, d_1 is the distance between point p_1 and p_1', d_2 is the distance between point p_2 and p_2'. So the *disCheck* is equals to 1 if $\sum(d\_1, d\_2)$ is smaller than a threshold, which means tripleg segmentation successfully, 0 if not and *ed* is the edit distance calculated by similarity measures.

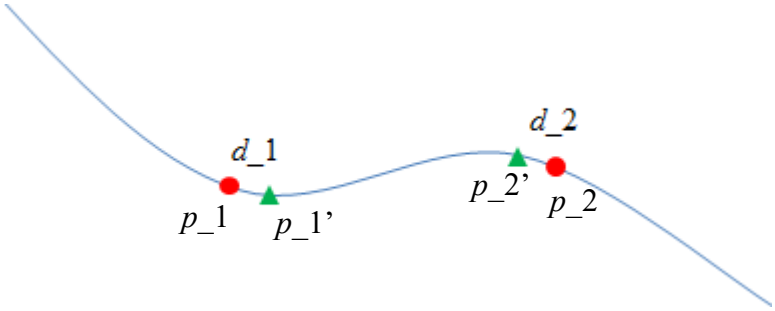

Figure 3.6: Example of the *disCheck*, two dots are the real segmenting points and two triangles are the segmenting points based on the similarity measures, d_1 is the distance between point p_1 and p_1', d_2 is the distance between point p_2 and p_2'

Considering about the evaluation of the tripleg segmentation, for coverage evaluation, firstly, the value of the edit distance (*ed*) needs to be length-normalized (here the number of locations contained in the trip is used as the length of the trip, because for edit distance calculation, the number of the location is regarded as the length of the trajectory). Then count the number of the trips whose length-normalized *ed* is smaller than a threshold ε as *num_cov_tripleg*. The ε is set as different values (0.1, 0.2 and 0.3). The total number of trips is *n*. Then the coverage of tripleg segmentation *cov_tripleg* can be defined as Equation 3.2:

$$cov\_tripleg = num\_cov\_tripleg / n \tag{3.2}$$

For accuracy, it is the percentage of the trips which implement tripleg segmentation correctly. If *num_acc_tripleg* is the number of trips whose *disCheck* equals to 1 and *num_tripleg1* equals to *num_tripleg2* in Table 3.2. Then the accuracy of the tripleg segmentation *acc_tripleg* can be defined as Equation 3.3. The Algorithm 10 in Appendix implements the idea of it.

$$acc\_tripleg = num\_acc\_tripleg / num\_cov\_tripleg \tag{3.3}$$

3.5 Trajectory similarity based travel mode inference

With the accomplishment of the tripleg segmentation, the next step is to find the transportation mode used on each tripleg. Mode inference is also based on similarity

measures with the historical labeled trip. After tripleg segmentation, a tripleg *l* of a user *u* is selected and is compared against every tripleg *l_i* in *H_L*, where *H_L* is a historical set of labeled trip legs of *u*. Here the travel mode information of the triplegs is not just projected based on the first match that is used to identify the boundaries of triplegs, because some triplegs' length are relatively short compared with the whole trip's length and even if two trips have a low value of edit distance (which means they are similar trips), they may have different travel modes at that short distance tripleg. For example, in Figure 3.7, this user has three historical trips. Trip_1 (green line) contains tripleg_1 (bicycle, 0.6km) and tripleg_3 (bus, 15km); trip_2 (blue line) contains tripleg_2 (0.4km, walk) and tripleg_4 (bus, 15km); trip_3 (black line) contains tripleg_5 (0.3km, walk). Now, the trip_2 is regarded as unlabeled trip. Based on the similarity measure based tripleg segmentation, the trip_1 and trip_2 are the most similar trips and trip_2 will have the same tripleg segmentation with trip_1. However, if the method just projects the mode distribution of the trip_1 to trip_2, the tripleg_2 of trip_2 will be inferred falsely. Therefore, the tripleg_2 of the trip_2 should be selected out and be compared with all the other historical triplegs so that the tripleg_5 can be found to be the most similar tripleg of tripleg_2 and apply the mode of tripleg_5 (walk) to the tripleg_2, which makes the mode inference exactly.

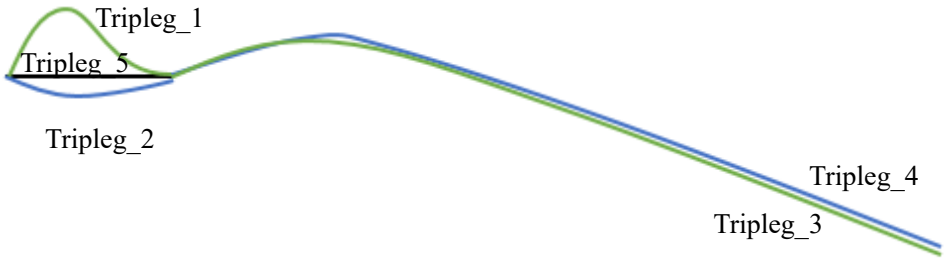

Figure 3.7: Motivation for trajectory similarity based travel mode inference.

For mode inference, assume each tripleg in the data set as a new unlabeled user's tripleg, the compare it with all his/her historical triplegs based on similarity measure and return the most similar tripleg's id and the value of edit distance between them as *sim_tripleg[a, b]* in Table 3.3, which *sim_tripleg[a]* is the similar tripleg's id and *sim_tripleg[b]* is the value of the edit distance. Then just apply the type of travel mode of the most similar triplegs to new user's triplegs. The Algorithm 11 in Appendix shows this procedure.

Evaluating the mode inference performance is also based on the same indexes: coverage and accuracy. Features used are shown in Table 3.3.

Table 3.3: List of features used by mode inference performance evaluation

Tripleg_id	Id of the identified tripleg
user_id	id of the user
sim_tripleg[*a*,*b*]	the most similar tripleg's id *a* and the edit distance *b*
num_location	number of locations contained in this tripleg
duration	time interval of the tripleg
travel_mode1	the real type of the travel mode of the identified tripleg
travel_mode2	the type of the travel mode of the most similar tripleg
mode_eva	mode inference successfully or not

Specially, *mode_eva* is a Boolean indicator (1 if *travel_mode1=travel_mode2,* 0 if else).

For coverage evaluation, firstly, length-normalized the value of the edit distance *sim_tripleg[b]* (here the number of locations contained in the tripleg is also used as the length of the trip same with the tripleg segmentation just like tripleg segmentation). Then count the number of the triplegs whose length-normalized *sim_tripleg[b]* smaller than the threshold ε (different values with 0.1, 02 and 0.3) as *num_cov_mode*. The total number of triplegs is *m*. Then the coverage of mode inference *cov_mode* can be defined as Equation 3.4:

$$cov\_mode = num\_cov\_mode / m \tag{3.4}$$

For accuracy, it is the percentage of the trips which achieve mode inference correctly and if the *num_acc_mode* represents the number of triplegs whose *mode_eva* equals to 1. The accuracy of mode inference is defined as Equation 3.5. The Algorithm 12 in Appendix implements the idea of it.

$$acc\_mode = num\_acc\_mode / num\_cov\_mode \tag{3.5}$$

4 Experiments

4.1 Data collection system and database schema

The data used in this thesis is the manually labeled trip information (route, mode etc.) and the raw data (accelerometers and location tracks) collected by MEILI system. MEILI system is the main tool to collect data in this project. MEILI is a travel diary collection, annotation and automation system. In MEILI, a user installs an application that collects his / her GPS trace which then the user subsequently annotates with trip and activity information via a web GIS based MEILI web application (Prelipcean 2015). MEILI system automatically collects raw data, drivers' features and performs predictions for the three basic tasks: trip segmentation, tripleg segmentation and mode

inference (as well as the tasks of destination location inference and trip purpose inference). It also provides a GUI (Graphical User Interface) for the users to view and verify / correct the data and the inferences, since it cannot be guaranteed that the inferences it makes are correct.

Consideration of Privacy Laws and Regulation, in this thesis, all the personal data, like the manually labeled trip information (route, mode, etc.) and the raw data (accelerometers and location tracks) collected by the MEILI system are used and analyzed under strict legal conditions and with a specific purpose, which is never to identify an individual. More specifically, the privacy rights of the data owner (i.e., the individual whose trips are recorder in the data) are respected. In particular, during the research and the publication of the results, all privacy laws and regulations (like the General Data Protection Regulation (Presidency of the Council 2015)) are followed.

4.2 Overview of the dataset

The dataset used in this thesis contains the following sections: GPS locations which contains the information about the latitude and the longitude of the point, time, speed, locations' id, users' id, etc., trips with information about trips' id, user's id who takes the trip, start point, end point, start time, end time, number of triplegs, purpose, etc., triplegs with information about trips' id, user's id who takes the trip, start point, end point, start time, end time, transportation mode of this tripleg, etc. The Table 4.1 is the overview of the dataset collected for this thesis.

Table 4.1: Overview of the dataset conducted for this thesis

Start date	02.11.2015
End date	19.04.2016
Number of users	206
Number of GPS locations	1012229
Number of trips	4240
Number of triplegs	5431
Number of travel mode	14

Specially, the 14 types of mode are showed in Table 4.2.

Table 4.2: Overview of the 14 types of transportation mode

Id	Name	Number of triplegs
1	Walk	2012
2	Bicycle	319
3	Moped / Motorcycle	44
4	Car as driver	899
5	Car as passenger	256
6	Taxi	419
7	Bus	462
8	Subway	381
9	Tram	202
10	Commuter train	217
11	Train	120
12	Ferryboat	60
13	Flight	33
14	Other	7

4.3 Test of similarity measure

Before implementing the tasks of tripleg segmentation and mode inference, it is necessary to test the similarity measure. Here the matching threshold ε of EDR distance mentioned in Section 3.2 is set as 100m (nearly 0.001°). A random user's trip is chosen to test the measure, showed in Figure 4.1. The distribution information of the values of most similar edit distance of this user's all trips are showed in Table 4.3.

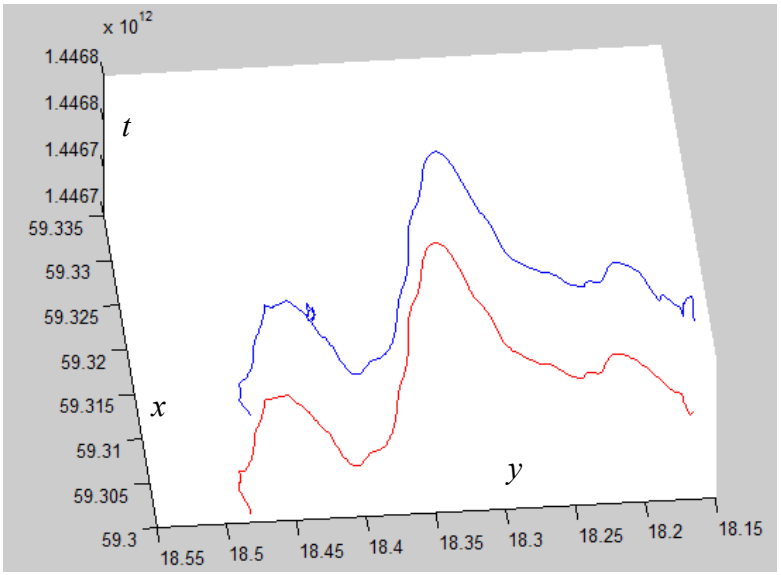

Figure 4.1: The result of the test of similarity measure, the above line is the test trip and the bottom line is the most similar trip find by EDR similarity measure. *x, y* is the location of this user, *t* is the real time of the location, expressed in timestamp / ms.

Table 4.3: The statistical information of the values of most similar edit distance of this user's all trips

indexes	edit distance	length-normalized edit distance	number of locations contained in the trip
average	33.56	0.075	72
min	0	0	2
max	240	1	429
standard deviation	56.87	0.125	113

It can be seen form the Figure 4.2 intuitively that the two trajectories are extremely similar with low value of EDR distance (EDR=20 with 350 points contained in these two trajectories, 20/350=0.057<0.1). With the help of value of average and standard deviation of length-normalized edit distance in Table 4.3, it is better to judge what the similarity based matching threshold ε should be. Here the ε can be set around 0.2 (0.075+0.125=0.2).

4.4 Results evaluation and discussion

After the data processing, the result of each task can be obtained. As mentioned, the analysis and evaluation focus on two indexes: coverage and accuracy.

4.4.1 Trip segmentation

As mentioned in Section 3.2, when doing accuracy measure, a buffer size Δd is taken into considered. And the buffer size has been varied to see its influence. The result of the trip segmentation can be seen in Table 4.4.

Table 4.4: The trip segmentation accuracy measure results vary with the buffer size, where "*buffer size*" is the width of the buffer area (relative to the length of the trip). "*start / end points detected*" is the number of the trip's start / end point which has been detected successfully, "*trips detected*" is the number of trips whose both start point and end point are detected successfully, "*total number of trips*" is the number of trips in dataset whose length satisfy the minimum distance" and "*accuracy*" is percentage of trips detected, obtained by *(trips detected) / (total number of trips).*

buffer size	start points detected	end points detected	trips detected	total number of trips	accuracy
0.1	824	837	660	4240	15.56%
0.2	1357	1324	1023	4240	24.13%
0.3	2002	1942	1629	4240	38.42%

It can be seen from Table 4.4 that with increase of the buffer size, the accuracy also

increases. It is easy to understand that with the buffer size increase, more start / end points can be detected in the buffer area and as a result, more trips can be detected. However, this buffer size cannot be too large, because it increases the error tolerance range too much and makes the results meaningless. Moreover, when doing trip segmentation, it is found that for longer distance trips, they can be detected more accurate than short distance trip, so it seems the minimum length has a large influence on the accuracy. As a result, different values of the minimum length have been set to see its effects on accuracy. The Table 4.5 shows the results. (0.2 is the buffer size for this evaluation)

Table 4.5: The evaluation result of trip segmentation compared with the minimum length of the trip, where "*minimum length*" is requirement of the minimum length of the trip.

minimum length / km	start points detected	end points detected	trips detected	total number of trips	accuracy
0	1357	1324	1023	4240	24.13%
5	757	742	482	1490	32.30%
10	590	606	402	1089	36.91%
25	359	347	311	514	60.51%
50	145	130	127	194	65.46%
80	81	73	67	94	71.28%

It can be obtained from the Table 4.5 that the accuracy increases from 24% for all trips to 70% for trips whose length is larger than 80km. When the minimum length is larger than nearly 20km, the accuracy can reach up to 55%. This indicates that for the dataset used in this experiment, the longer the trip's length is, the larger accuracy of the trip segmentation by using the heuristic rule will be. This is because when length smaller than 10km, there will be many walking periods contained in the trip and the speed of walk is too small for GPS recording, lots of walking-periods' speed have been recorded smaller than 1 m/s. As a result, this will definitely influence the detection when speed is an important factor for the heuristic rule. One of the future research directions is to investigate how to increase the accuracy of the low-distance-trip based on the heuristic rule.

4.4.2 Tripleg segmentation

Some relevant statistics about the characteristics of the trips and triplegs are shown in Table 4.6 and Table 4.7.

Table 4.6: Relevant statistics about trips

Trips	number of trips	avg of number of triplegs per trip	std of number of triplegs per trip
length>0km	4078	1.51	1.43
length>10km	1089	2.42	2.28

Table 4.7: Relevant statistics about triplegs

Triplegs	number of triplegs	top 1 mode by distance	top 2 mode by distance
length>0km	5464	Car as driver	Bus
length>5km	1049	Car as driver	Subway

For tripleg segmentation task, as mentioned in Section 3.4, different values (0.1, 0.2 and 0.3) have been set as the threshold ε. As a reminder, ε is a minimum distance based matching threshold and points' 3D length-normalized edit distance of two trajectories beyond this threshold are not matched and incur a penalty. This means that a higher value of threshold entails a fuzzier matching. Two indexes (coverage and accuracy) are evaluated and the results are showed in Figure 4.2 (coverage) and 4.3 (accuracy). Here the x-variable for calculating accuracy and coverage is the percentage of the datasize, which means the percentage of the trips / triplegs which are taken into consideration to find similar trips / triplegs from historical diaries. For example, in all figures of this section, the 20% of datasize means the first 20% of the all trips according time order.

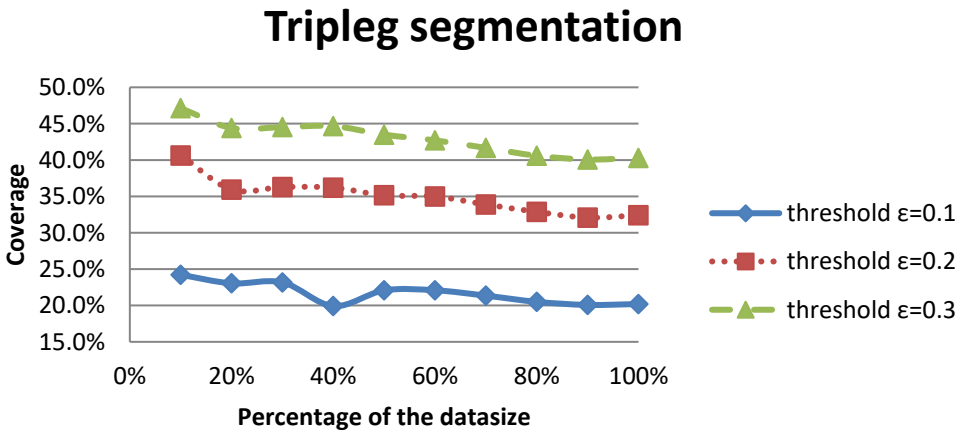

Figure 4.2: The coverage of the tripleg segmentation result, which means the percentage of the inferable trips compared with the percentage of the datasize, which is defined by length-normalized edit distance smaller than the threshold.

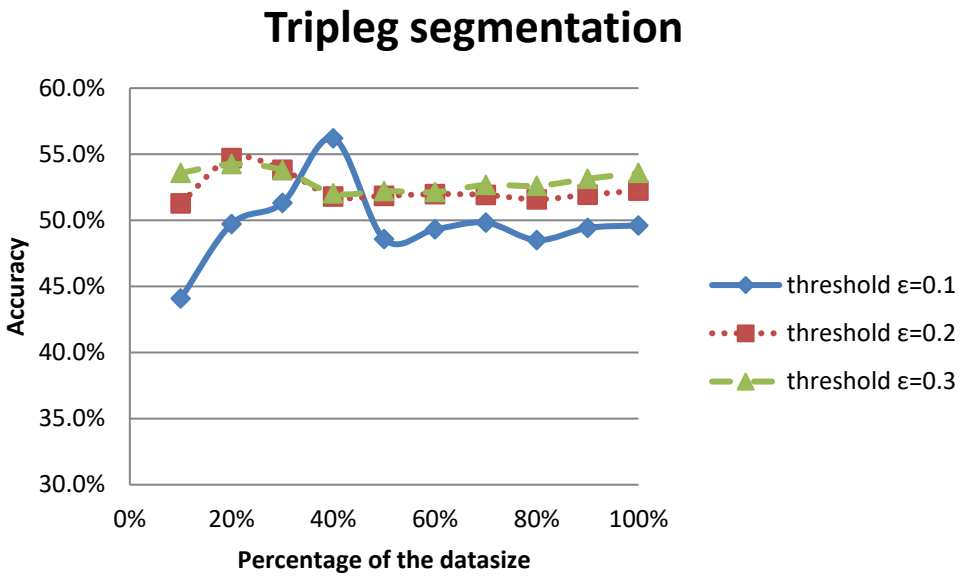

Figure 4.3: The accuracy of the tripleg segmentation result, which means the percentage of the trips which achieve correct tripleg segmentation successfully.

Firstly, it can be obtained from the Figure 4.3 that with the increase of the value of threshold, more inferable trips can be got. Specially, coverage goes stably around 20% with the increase of the datasize when ε is 0.1, nearly 32.5% for ε as 0.2 and about 40% for ε equals to 0.3. It is easy to understand that when increase the value of threshold, there can be more trips satisfying the condition of length-normalized edit distance smaller than the threshold. For accuracy, it is stable around the 50% for ε as 0.1, 52% when ε equals 0.2, and 53% for ε as 0.3 which has the highest value of accuracy. In general, a decreasing trend can be seen for coverage, which is kind of counter-intuitive, because with increased history size one expects that the method will be able to find a trip that matches a new unlabeled trip with increasing probability. In particular, let $C(x)$ and $H(x)$ represents the coverage and the size of the history that is based on the first x% of the trips. Then intuitively, some trips / triplegs which do not belong to $H(x)$ but are in $H(y)$ have been covered by $H(x)$ and are not taken into consideration when calculating $C(y)$ ($y>x$). However, this is to the case because in Figure 4.2 and 4.3, there is an uneven change at 40% of the datasize when ε is 0.1 can be explained by this situation. So more data may be needed to get an increasing trend.

Obviously, the accuracy is not satisfactory enough. With the inspiration of trip segmentation evaluation, the length of the trajectory may also influence the accuracy of the tripleg segmentation. So the trips whose length is larger than 10 km (nearly 0.1°) are selected and do tripleg segmentation again. The results are showed in Figure 4.4 and 4.5.

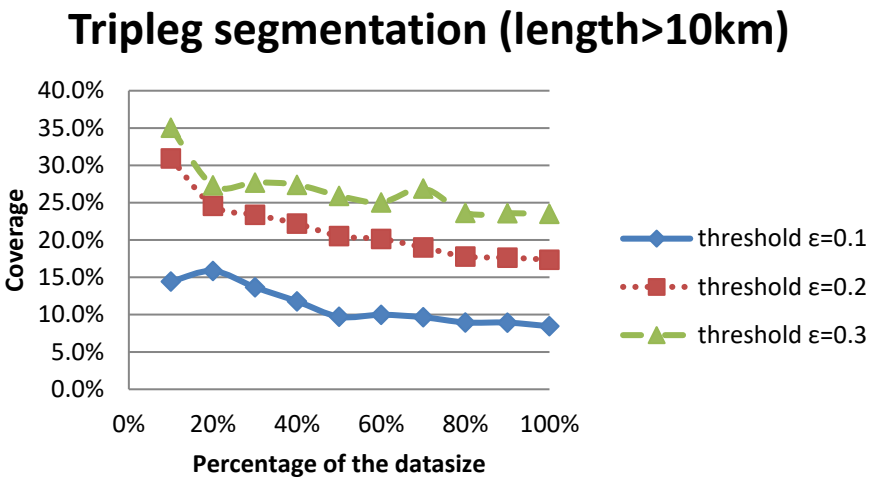

Figure 4.4: The coverage of the tripleg segmentation result with different values of threshold when the minimum length of the all trips is larger than 10 km

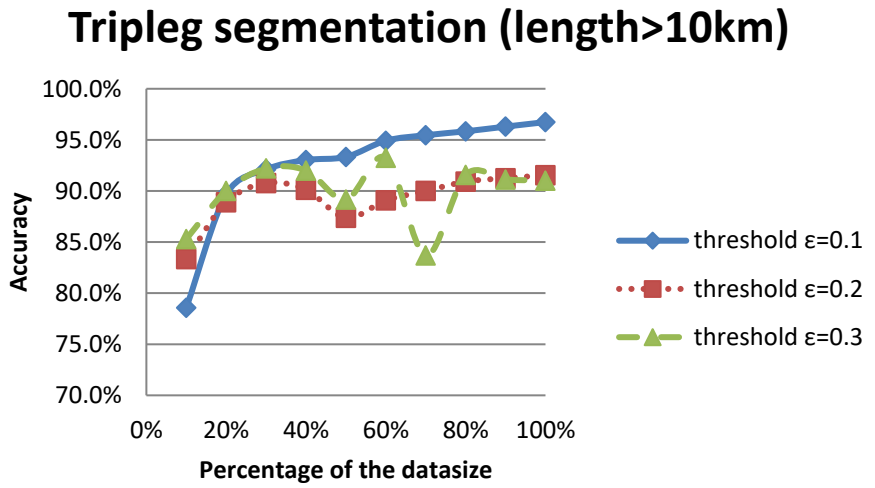

Figure 4.5: The accuracy of the tripleg segmentation result with different values of threshold when the minimum length of the all trips is larger than 10 km.

It can be seen from Figure 4.5 that the accuracy has increased a lot; especially when ε is 0.1, the accuracy is around 97% with all data used. It increases 47% compared with the first condition (no minimum length limitation). For ε with 0.2 and 0.3, they also share the dramatic growth: 91.5% increased by 39.5% for threshold as 0.2 and 91% rise by 38% with ε as 0.3. So the same with trip segmentation, this similarity measure and historical data based tripleg segmentation is more accurate for long-distance-trip (larger than 10km). However, the curve with 0.3 as threshold is uneven compared with the other two. It has an apparent bottom point when temporally 70% of trips are considered. This is because the random fluctuations similar with the uneven changes in Figure 4.2 and 4.3. Moreover, different with no length limitation, the accuracy decreases, but not a lot, with increasing the values of threshold. However, neither of the two conditions has a high value of coverage, this is because the limitation of each

user's travel diary data. The data set has 4240 trips with 206 users, which means that averagely each user only has 20 trips for his / her travel diary database, while in real life; this number is too few for a person. As a result, many trips cannot find a similar historical trip and this leads a low value of the coverage.

Here taking the tripleg segmentation as the example, the effects of taking reverse trip into consideration are evaluated as mentioned in Section 3.3. Firstly, set ε equals to 0.2 and no minimum length limitation, and then evaluate the tripleg segmentation again based on the similarity measure which cannot match a labeled historical trip to the reverse of a trip. The results are shown in Figure 4.6 (coverage) and 4.7 (accuracy).

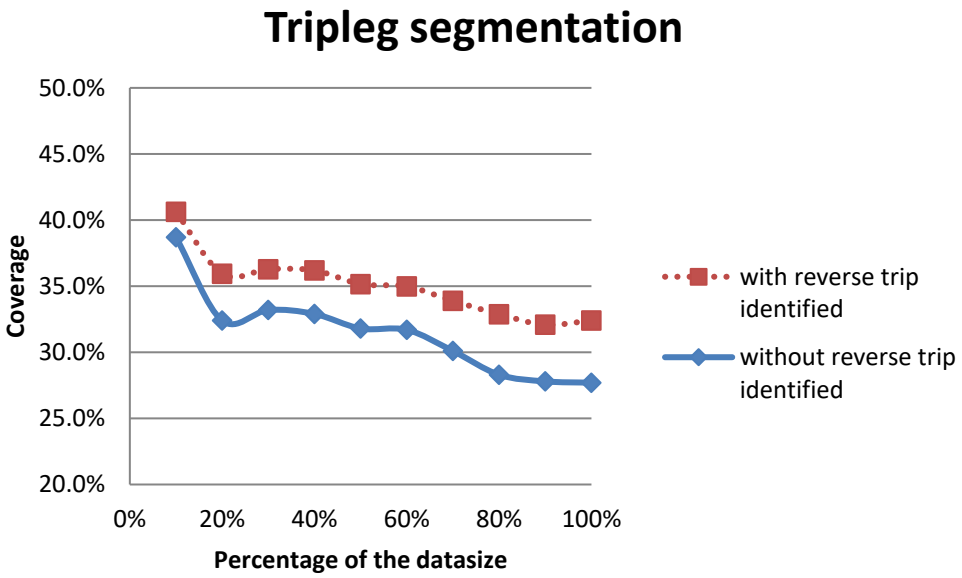

Figure 4.6: The coverage evaluation results of the effect of reverse trip identifiable based on tripleg segmentation

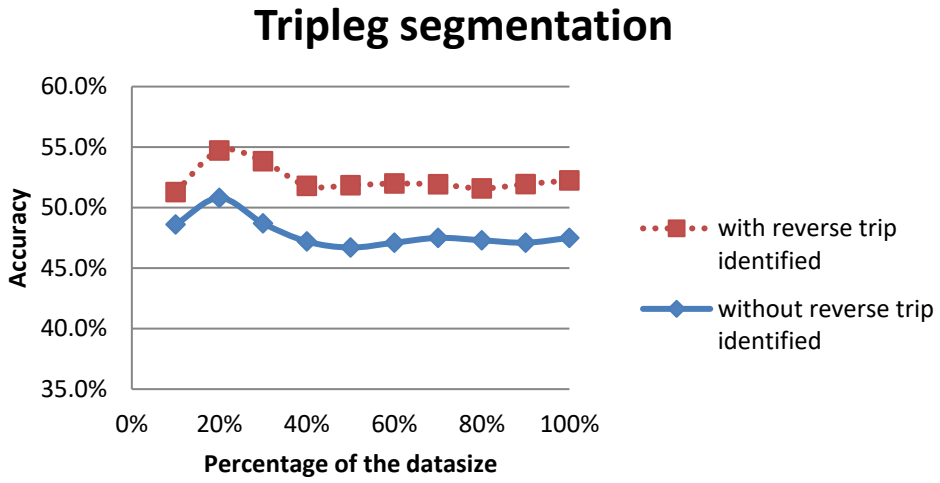

Figure 4.7: The accuracy evaluation results of the effect of reverse trip identifiable based on tripleg segmentation

It can be seen from Figure 4.6 and 4.7 that both coverage and accuracy have increased by nearly 10% because of the reverse trip identifiable and the benefit of this feature can be seen obviously.

4.4.3 Mode inference

For mode inference task, as mentioned in Section 3.5, different values (0.1, 0.2 and 0.3) also have been set as the threshold of the length-normalized edit distance to define the inferable triplegs. Two indexes (coverage and accuracy) are evaluated and the results are showed in Figure 4.8 (coverage) and 4.9 (accuracy). In all figures of this section, the 20% of datasize means the first 20% of the triplegs according time order.

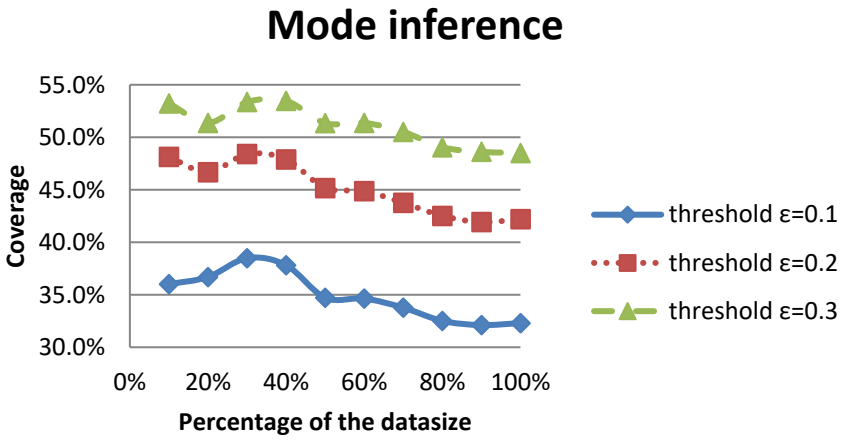

Figure 4.8: The coverage of mode inference result with different values of threshold

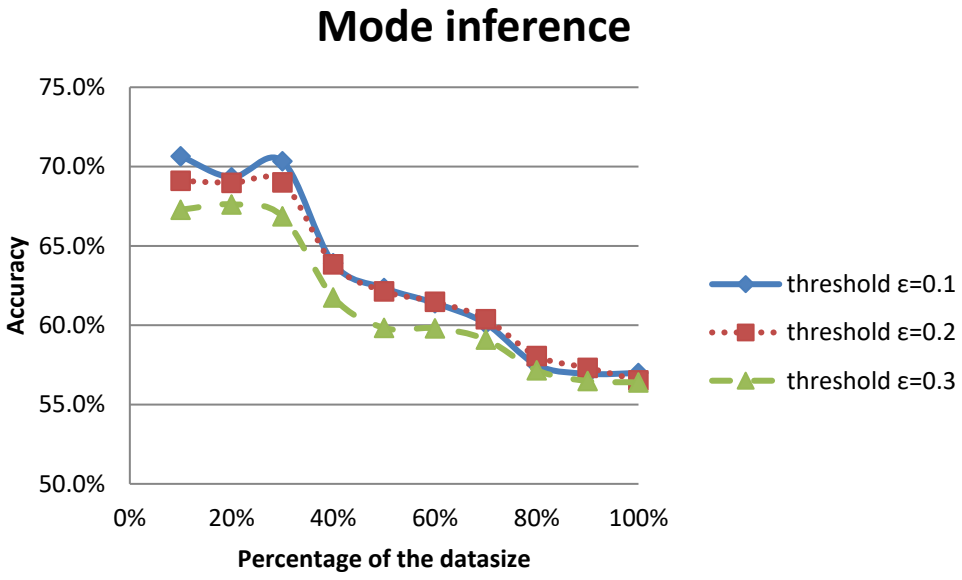

Figure 4.9: The accuracy of mode inference result with different values of threshold.

As showed in Figure 4.8, the coverage of the result of mode inference also increases with the growth of the threshold with the same reason of the tripleg segmentation but overall it is larger than the results of tripleg segmentation because of the larger number of triplegs than trips'. However, in Figure 4.9, the accuracy decreases, but not a lot, with increasing the values of threshold. The reason for the accuracy decrease is that a high standard for two trajectories to be similar will decrease the number of inferable trips but increase the possibilities that the two trajectories have the same transport mode sequence / segmentation. The highest one is 57% considering the all triplegs when ε is 0.1 and the lowest one is 56.4% for ε as 0.3. Overall, the accuracy trends to between 56% and 58%. The decreasing trend of the accuracy also depends on the order of the datasize and has kind of occasionality.

Similarly, the length of the triplegs is also taken into consideration for mode inference and the result is showed in Figure 4.10 and 4.11.

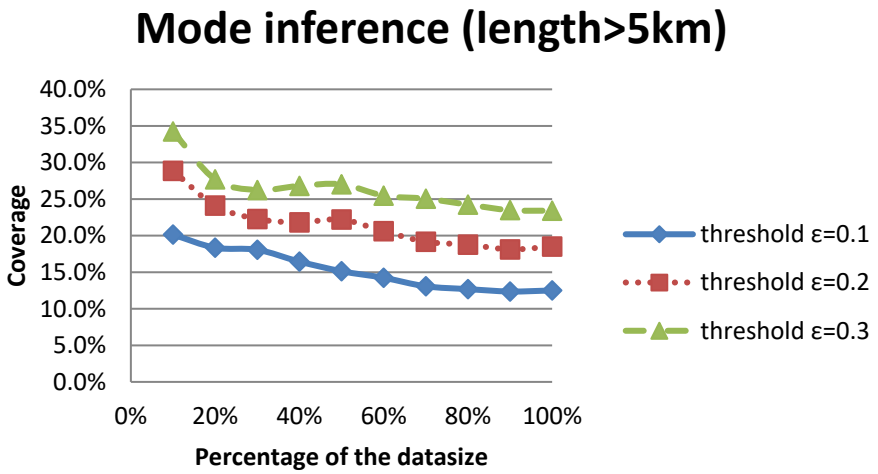

Figure 4.10: The coverage of the mode inference result with different values of threshold when the minimum length of the all trips is larger than 5 km.

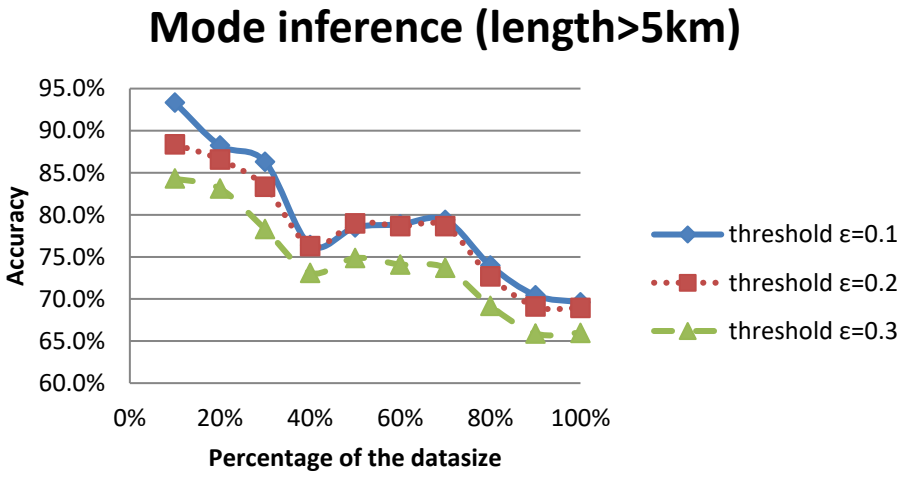

Figure 4.11: The accuracy of the mode inference result with different values of threshold when the minimum length of the all trips is larger than 5 km.

Here the length threshold is 5km (nearly 0.05°), smaller than tripleg segmentation, because generally the length of the triplegs is smaller than trips'. Apparently, compare the Figure 4.11 with 4.9, an obvious increase of accuracy can be observed. The accuracy now is between 66%~70%, increased by nearly 10%. Similarly, the accuracy also decreases, but not a lot, with increasing the values of ε. However, the decreasing trend here is more dramatic compared with Figure 4.9. This is because that firstly, the number of triplegs whose length larger than 5 km is 1534, much smaller than the total number of triplegs 5431, which means that one tripleg's mode inference failed has a larger effect on accuracy of long distance trips than short ones'. Secondly, many long distance trips are only done once by the user (like flight and train) and it is hard to find a similar tripleg in user's historical tripleg dataset (low coverage), as a result, thinking about these two factors, when the number of considered triplegs are too few, the accuracy of mode inference decreases a lot and it becomes stable until nearly all triplegs are taken into consideration.

A confusion matrix for the mode detection task is showed in Table 4.8.

Table 4.8: Confusion matrix for the mode detection task with the condition: ε is 0.1. The diagonals data are the number of triplegs which have been correctly classified for the specific mode and the accuracy. The other data are the number of triplegs which have been confused by other specific mode when doing mode inference.

	Actual class														
predicted class	mode	Walk	Bicycle	Moped / Motorcycle	Car as driver	Car as passenger	Taxi	Bus	Subway	Tram	Commuter train	Train	Ferryboat	Flight	Total
	Walk	596 (69.1%)	22	2	28	14	33	14	13	7	11	7	0	0	862
	Bicycle	20	45 (54.9%)	0	5	1	2	1	0	0	4	0	0	0	82
	Moped / Motorcycle	3	0	7 (43.8%)	3	0	1	0	0	0	0	0	2	0	16
	Car as driver	37	4	1	102 (63.8%)	6	8	3	1	6	1	0	2	0	158
	Car as passenger	12	2	0	7	30 (54.5%)	4	7	0	0	0	0	0	0	55
	Taxi	48	3	1	9	7	86 (61.4%)	3	1	2	0	1	0	0	139
	Bus	22	3	0	0	4	2	69 (60.5%)	3	0	1	3	1	0	114
	Subway	7	0	0	2	0	1	3	30 (69.8%)	1	0	0	0	0	43
	Tram	10	1	0	5	0	2	1	1	29 (65.9%)	1	0	0	0	44
	Commuter train	10	2	0	1	0	0	2	1	1	28 (65.1%)	1	0	0	43
	Train	5	0	0	0	0	3	1	0	1	0	13 (52%)	0	0	25
	Ferryboat	5	0	0	0	0	0	0	0	0	0	0	8 (61.5%)	0	13
	Flight	0	0	0	0	0	0	0	0	0	0	0	0	4 (100%)	4

It can be seen from Table 4.8 that subway and walk are mostly correctly predicted relatively. Moreover, walk is often confused with taxi, car and bicycle. Moped is often confused with walk and car. The rest modes are often confused with walk except for

flight. Most of the triplegs confused with walk are extremely short triplegs with low speed. This may also explain the increase in accuracy for longer trips. It is strange that walk is confused with motorized transport (like car and taxi). The reason for this strange class confusion is unknown and can be the subject of future work. Some other feature based methods, see Prelipcean (2016b and 2016c), have less trouble differentiating between these modes that have highly different motion characteristics, hence hybrid methods can be used in the future to correct these misclassifications.

Cohen's Kappa coefficient (κ) is used to summarize the classification results here. It is a statistic to measure the agreement between two raters when each of them classifies A items into B categories (Smeeton 1985). If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters, $\kappa \leq 0$. Here it is used to measure the consistency of actual class and predicted class. The κ for the confusion matrix is 0.52, which means the actual class has the moderate consistency with the predicted class.

To sum up the tripleg segmentation and mode inference, 1) in general, low value of the threshold (low value of length-normalized edit distance limitation) can obtain high accuracy but low coverage, because a high standard for two trajectories to be similar will decrease the number of inferable trips but increase the possibilities that the two trajectories have the same transport mode sequence / segmentation. The Figure 4.5 and 4.6 shows the different conclusion, because the coverage between them is much larger than other three situations; 2) all situations has a low value of coverage, this is because the limitation of each user's travel diary data and many trips / triplegs cannot find a similar historical trip / tripleg; 3) the decreasing or increasing trend of the coverage / accuracy depends on the order of the datasize and has kind of occasionality. So it does not have meaningful indications; 4) no matter tripleg segmentation or mode inference, they are both sensitive to the length of the trajectory. Long-distance-trips / tripleg have a higher value of accuracy than low-distance-trip / tripleg. This is probably because that the long-distance-trip has more regularities and people are not willing or able to break down these regularities (like the route followed, often using subway or train, car or public transport), they do not have many options in long-distance-trip. However, for low-distance-trips, people can have many options collocations, so even two trajectories are similar based on similarity measures (like the car and bus), and they may have different transport mode sequence / segmentation; 5) moreover, compared the results with the previous work, firstly, for the task of trip segmentation, Prelipcean (2016a) gains the accuracy of 28.6% for 2142 trips (in this case, 24.13% for 4240 trips with buffer size equals to 0.2), which has been a similar result. For tripleg segmentation, Schüssler et al., 2011 report a tripleg segmentation accuracy of 68%, while in this case, it is around 52%, but over 90% for trips' length is larger than 10 km, so it is necessary to increase the accuracy for short distance trip in future research. For mode inference, Prelipcean (2016a) performs the accuracy of 64.40% for 16 types of modes and 5961 triplegs with classifier type of Nearest Neighbor, compared with nearly 57.5% for 14 types of modes and 5431 triplegs

(around 68% for tripleg's length is larger than 5 km) in this thesis.

5 Conclusion

Travel diary data processing has become a hot research in recent years and GPS-based user trajectory similarity analysis is also a very important area. Based on the related research and the basic idea that a trip which has a (geographically and temporally) very similar route to a previously historical mode-segmented / labeled trip will likely have the same destination as well as transport mode sequence / segmentation, this dissertation puts forward a method of travel diary semantics enrichment of trajectories based on historical labeled data and trajectory similarity measures, which mainly includes the following tasks: 1) implement trip segmentation based on the heuristic rule: if the speed is very low (3.6 km/h used in this dissertation) for longer than a period of time (120s used in this thesis), then the end of a trip has been identified; 2) define the similarity measures between two trajectories based on the EDR distance; 3) implement tripleg segmentation based on similarity measures and the historical labeled data, then apply the same tripleg segmentation pattern of the most similar historical labeled trip to the new identified trip; 4) implement mode inference also based on similarity measures, then apply the transportation mode of the most similar historical tripleg to the new unlabeled tripleg. After the evaluation of the result, it can be obtained that this trajectory similarity measures and historical labeled data based method is valid, however, this method is sensitive to the length of the trajectory. The result shows a better performance when the length of the trajectory is relatively longer (larger than 10km).

6 Future work

Firstly, as mentioned in conclusion, the method proposed in this thesis is sensitive to the length of the trajectory. As a result, one important direction in the future is to study how to increase the accuracy of trip segmentation, tripleg segmentation and mode inference for low-length-trip. Secondly, in the thesis, all users use the same similarity measures but it does not take the individual needs into consideration. In real life, different users have different preferences about their own travel diary and using the same similarity measures for different users may not be able to meet diversities. So the next step is to study how to build a trajectory similarity measures easy to meet the different preferences of individual. Finally, with known of the travel diary of different users, it can infer the similarity between two users' behavior so that a user can be recommended to other users who has a high level of similarity. This can increase the chances of social contact between people.

7 References

Agrawal, R., Faloutsos, C., & Swami, A. N. (1993). Efficient Similarity Search In Sequence Databases. *International Conference on Foundations of Data Organization and Algorithms* (Vol.730, pp.69-84). Springer-Verlag.

Chen, L., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. *ACM SIGMOD International Conference on Management of Data* (pp.491-502). ACM.

Chen, L., & Ng, R. (2004). On the Marriage of Edit Distance and Lp Norms. *Very Large Data Bases*.

Crochemore, M., & Rytter, W. (1994). *Text algorithms*. Oxford University Press, Inc. ISBN 0-19-508609-0

Deng, K., Xie, K., Zheng, K., & Zhou, X. (2011). *Trajectory Indexing and Retrieval*. *Computing with Spatial Trajectories*. ISBN 978-1-4614-1628-9

Doherty, S. T., Noël, N., Gosselin, M. L., Sirois, C., and Ueno, M. (2001). Moving beyond observed outcomes: integrating global positioning systems and interactive computer-based travel behavior surveys. *Transportation Research Circular*.

Filip Biljecki, Hugo Ledoux, & Peter van Oosterom. (2013). Transportation mode-based segmentation and classification of movement trajectories. *International Journal of Geographical Information Science Ijgis,* 27(2): 385-407.

Gong Xudong. (2011). *Similarity Search of Trajectory data and Its Application* (Doctoral dissertation, Hefei: University of Science and Technology of China).

Hemminki, S., Nurmi, P., & Tarkoma, S. (2013). Accelerometer-based transportation mode detection on smartphones. *ACM Conference on Embedded Networked Sensor Systems* (pp.13). ACM.

K Kucuk, A., Hamdi, S. M., Aydin, B., Schuh, M. A., & Angryk, R. A. (2016, October). PG-TRAJECTORY: A PostgreSQL/PostGIS based Data Model for Spatiotemporal Trajectories. In *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on* (pp. 81-88). IEEE.

Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., & Ma, W. Y. (2008). Mining user similarity based on location history. *ACM Sigspatial International Conference on*

Advances in Geographic Information Systems (pp.34). ACM.

Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., ... & Theodoridis, Y. (2013). Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, 45(4): 42.

Prelipcean, A. C., Gidofalvi, G., & Susilo, Y. O. (2015). Comparative framework for activity-travel diary collection systems. *International Conference on MODELS and Technologies for Intelligent Transportation Systems* (pp.251-258). IEEE.

Prelipcean, A. C. (2016a). *Capturing travel entities to facilitate travel behaviour analysis: A case study on generating travel diaries from trajectories fused with accelerometer readings* (Doctoral dissertation, KTH Royal Institute of Technology).

Prelipcean, A. C., Gidofalvi, G., & Susilo, Y. O. (2016b). Measures of transport mode segmentation of trajectories. *International Journal of Geographical Information Science*, 30(9): 1763-1784.

Prelipcean, A. C., Gidófalvi, G., & Susilo, Y. O. (2016c). Transportation mode detection–an in-depth review of applicability and reliability. *Transport Reviews*, 37(4): 442-464.

Prelipcean, A.C., Gidófalvi, G. and Susilo, Y.O. (2016d). MEILI: A Travel Diary Collection, Annotation and Automation System. *Journal of Urban Technology*.

Presidency of the Council. (2015). "Compromise text. Several partial general approaches have been instrumental in converging views in Council on the proposal for a General Data Protection Regulation in its entirety. The text on the Regulation which the Presidency submits for approval as a General Approach appears in annex.", pages 201

Rasmussen, T. K., Ingvardson, J. B., Halldórsdóttir, K., & Nielsen, O. A. (2015). Improved methods to deduct trip legs and mode from travel surveys using wearable GPS devices: A case study from the Greater Copenhagen area. *Computers, Environment and Urban Systems*, 54: 301-313.

Sefidmazgi, M. G., Sayemuzzaman, M., & Homaifar, A. (2014). Non-stationary time series clustering with application to climate systems, 312: 55-63.

Schuessler, N., & Axhausen, K. W. (2009). Processing raw data from global positioning systems without additional information. *Transportation Research Record Journal of the Transportation Research Board,* 2105(2105): 28-36.

Shalaby, E. H. C. A. (2005). A trip reconstruction tool for gps-based personal travel

surveys. *Transportation Planning & Technology*, 28(5): 381-401.

Smeeton, N. C. (1985). Early history of the kappa statistic. Biometrics, 41(3):795-795.

Stenneth, L., Wolfson, O., Yu, P. S., & Xu, B. (2011). Transportation mode detection using mobile phones and GIS information. *ACM Sigspatial International Symposium on Advances in Geographic Information Systems, Acm-Gis 2011, November 1-4, 2011, Chicago, Il, Usa, Proceedings* (pp.54-63). DBLP.

Stopher, P., FitzGerald, C., & Zhang, J. (2008). Search for a global positioning system device to measure person travel. *Transportation Research Part C: Emerging Technologies*, 16(3): 350-369.

Toohey, K., & Duckham, M. (2015). Trajectory similarity measures. *SIGSPATIAL Special*, 7(1): 43-50.

Tsui, S., & Shalaby, A. (2006). Enhanced system for link and mode identification for personal travel surveys based on global positioning systems. *Transportation Research Record Journal of the Transportation Research Board*, 1972(1): 38-45.

Wang, S., Chen, C., & Ma, J. (2010). Accelerometer Based Transportation Mode Recognition on Mobile Phones. *Asia-Pacific Conference on Wearable Computing Systems* (pp.44-46). IEEE Computer Society.

Wolf, J. L. (2000). *Using GPS data loggers to replace travel diaries in the collection of travel data* (Doctoral dissertation, School of Civil and Environmental Engineering, Georgia Institute of Technology).

Yang, A. Y., Jafari, R., Sastry, S. S., & Bajcsy, R. (2009). Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence & Smart Environments*, 1(2): 103-115.

Ying, J. C., Lu, H. C., Lee, W. C., Weng, T. C., & Tseng, V. S. (2010). Mining user similarity from semantic trajectories. *ACM Sigspatial International Workshop on Location Based Social Networks* (pp.19-26). ACM.

Zheng, Y., Chen, Y., Li, Q., Xie, X., & Ma, W. Y. (2010). Understanding transportation modes based on gps data for web applications. *ACM Transactions on the Web,* 4(1): 1-36.

8 Appendix

Algorithm 1: creating a Trajectory object from a collection of TG_PAIRs.

```
DROP TYPE IF EXISTS tg_pair CASCADE;
CREATE TYPE tg_pair AS ( -- timestamp-geometry pair type
      t timestamp,
      g geometry
);

DROP TYPE IF EXISTS trajectory CASCADE;
CREATE TYPE trajectory AS (
      s_time TIMESTAMP,
      e_time TIMESTAMP,
      bbox GEOMETRY,
      sampling_interval INTERVAL,
      tr_data tg_pair[]);

DROP FUNCTION IF EXISTS _trajectory(tg_pair[]) CASCADE;
CREATE OR REPLACE FUNCTION _trajectory(tg_pair[]) RETURNS trajectory AS
$BODY$
DECLARE
   t trajectory;
BEGIN

      t.bbox = tg_mbr($1);
      t.e_time = tg_end_time($1);
      t.s_time = tg_start_time($1);
      t.tr_data = array_sort($1);
      IF array_length($1, 1) > 1 THEN
            t.sampling_interval = (t.e_time - t.s_time) / (array_length($1, 1) - 1);
      ELSE
            t.sampling_interval = INTERVAL '-1 seconds';
      END IF;
      RETURN t;
END
$BODY$
LANGUAGE 'plpgsql';
```

Algorithm 2: calculate the number of the locations contained in the trajectory

```
DROP FUNCTION IF EXISTS t_length( tg_pair[] );
CREATE OR REPLACE FUNCTION t_length(tg tg_pair[])
   RETURNS INTEGER AS
$BODY$
```

```
DECLARE
   time_count           INTEGER;
   tgp                    tg_pair;

BEGIN

   if tg ISNULL THEN
      RETURN 0;
   END IF;

   time_count = 0;

   FOREACH tgp IN ARRAY tg
   LOOP
         time_count = time_count + 1;
   END LOOP;

   RETURN time_count;

END

$BODY$
LANGUAGE 'plpgsql';
```

Algorithm 3: calculate the length of the trajectory

```
DROP FUNCTION IF EXISTS t_distance( trajectory );
CREATE OR REPLACE FUNCTION t_distance(tr trajectory)
   RETURNS FLOAT AS
$BODY$

DECLARE
   length             FLOAT;
   tgp tg_pair;
   prev    tg_pair;

BEGIN
   if tr.tr_data ISNULL THEN
      RETURN -1;
   END IF;

   length = 0;

   prev = tg_head(tr.tr_data);
```

```
   FOREACH tgp IN ARRAY tr.tr_data
   LOOP
        length =    length + st_distance(prev.g, tgp.g);
        prev=tgp;
   END LOOP;

   RETURN length;

END

$BODY$
LANGUAGE 'plpgsql';
```

Algorithm 4: classify the state of each point into stops or moving

```
DROP FUNCTION IF EXISTS stop_detection(id int) CASCADE;
CREATE FUNCTION stop_detection(id int) RETURNS int AS $$
DECLARE
v double precision;
v1 double precision;
i double precision;
j int;
a int;
t bigint;
t1 bigint;
k double precision[];

      BEGIN
      SELECT speed into v from locations where id_location=id;
      SELECT time into t from locations where id_location =id;
      SELECT user_id into j from locations where id_location =id;
      t1=t+120000;
      IF v>1 THEN
            a=1;
      ELSIF v=-1 THEN
            a=-1;
      ELSE
            SELECT ARRAY_AGG(speed) into k FROM locations WHERE time
BETWEEN t AND t1 AND user_id=j;
            FOREACH v1 in ARRAY k
            LOOP
              i=v1-1;
                IF i<=0 THEN
                    a:=0;
                ELSE
```

```
            a:=1;
        END IF;
        EXIT WHEN i>0;
    END LOOP;

END IF;
return a;
END
$$ LANGUAGE plpgsql;
```

Algorithm 5: classify the stop point into start point.

```
DROP FUNCTION IF EXISTS trip_segmentation_start(id int) CASCADE;
CREATE FUNCTION trip_segmentation_start(id int) RETURNS int AS $$
DECLARE
i int;
j int;
k int;
a int[];
c int;
b int;
e int;

BEGIN
RAISE NOTICE   'times here is %', id;
SELECT stop_detection into b from locations where id_location =id;
IF b=1 THEN
SELECT user_id into i from locations where id_location =id;
j=id-5;
k=id+5;
SELECT ARRAY_AGG(stop_detection) into a FROM locations WHERE
id_location BETWEEN j AND id-1;
FOREACH c in ARRAY a
LOOP
        IF c=0 THEN
            e:=1;
        ELSE
            e:=0;
        END IF;
        EXIT WHEN c!=0;
    END LOOP;
return e;
ELSE
return 0;
END if;
```

```
END
$$ LANGUAGE plpgsql;
```

Algorithm 6: classify the stop point into end point.

```
DROP FUNCTION IF EXISTS trip_segmentation_end(id int) CASCADE;
CREATE FUNCTION trip_segmentation_end(id int) RETURNS int AS $$
DECLARE
i int;
k int;
b int[];
c int;
d int;
e int;

    BEGIN
    RAISE NOTICE    'times here is %', id;
    SELECT stop_detection into c from locations where id_location =id;
    IF c=1 THEN
    SELECT user_id into i from locations where id_location =id;
    ELSE
    k=id+5;
    SELECT ARRAY_AGG(stop_detection) into b FROM locations WHERE
    id_location BETWEEN id+1 AND k;
        FOREACH d in ARRAY b
        LOOP
            IF d=0 THEN
                e:=-1;
            ELSE
                e:=0;
            END IF;
            EXIT WHEN d=1;
          END LOOP;
    return e;
    ELSE
    return 0;
    END IF;
    END
    $$ LANGUAGE plpgsql;
```

Algorithm 7: check the success of the trip segmentation

```
DROP FUNCTION IF EXISTS trip_evaluation(id int) CASCADE;
CREATE FUNCTION trip_evaluation(id int) RETURNS int AS $$
DECLARE
i int;
```

```
j int;
a int[];
b int;
c int;
k tg_pair[];
m int;

    BEGIN
    SELECT (t).tr_data into k from trips where to_point_id=id;
    m=t_length(k);
    i=id-ceil(m/5);
    j=id+ceil(m/5);
SELECT ARRAY_AGG(end_point) into a FROM locations WHERE location_id
BETWEEN i AND j;
    FOREACH b in ARRAY a
    LOOP
            IF b=0 THEN
                c:=0;
            ELSE
                c:=1;
            END IF;
            EXIT WHEN b=1;
        END LOOP;
        return c;
    END
    $$ LANGUAGE plpgsql;

DROP FUNCTION IF EXISTS detect(id int) CASCADE;
CREATE FUNCTION detect(id int) RETURNS int AS $$
DECLARE
i int;
j int;
a int;

    BEGIN
        SELECT detect_start into i from trips where trip_id=id;
        SELECT detect_end into j from trips where trip_id =id;
        IF i=1 and j=1 THEN
          a=1;
          ELSE
          a=0;
          END IF;
          return a;
    END
```

```
$$ LANGUAGE plpgsql;
```

Algorithm 8: key part of the similarity measure

```
1) DROP FUNCTION IF EXISTS t_edit_distance( tg_pair[], tg_pair[], NUMERIC );
    CREATE OR REPLACE FUNCTION t_edit_distance(tg1 tg_pair[], tg2 tg_pair[], e NUMERIC)
      RETURNS FLOAT AS
    $BODY$
    DECLARE
      D int[][];
      v int;
      m INT;
      n INT;
      geom1 GEOMETRY;
      geom2 GEOMETRY;
      subcost INT;
      te TEXT;
    BEGIN

      m := t_length(tg1);
      n := t_length(tg2);

      --RAISE NOTICE 'i: %', m;

      D := array_fill(0, ARRAY[m, n]);

      FOR i IN 2..m LOOP
        D[i][1] := n;
      END LOOP;

      FOR j IN 2..n LOOP
        D[1][j] := m;
      END LOOP;

      FOR i IN 2..m LOOP
        FOR j IN 2..n LOOP

          geom1 = (tg1)[i].g;
          geom2 = (tg2)[j].g;


          subcost = 1;
          if edit_match(geom1, geom2, e) THEN
```

```
            subcost = 0;
          END IF;

D[i][j] := LEAST(LEAST(D[i-1][j-1] + subcost, D[i-1][j] + 1), D[i][j-1] + 1);

          --RAISE NOTICE 'i: %, j: %, D %', i, j,    D[i][j];

        END LOOP;
      END LOOP;

      RETURN D[m][n];

    END
    $BODY$
    LANGUAGE 'plpgsql';

2) DROP FUNCTION IF EXISTS edit_match( GEOMETRY, GEOMETRY, NUMERIC );
    CREATE OR REPLACE FUNCTION edit_match(g1 GEOMETRY, g2 GEOMETRY, e NUMERIC)
      RETURNS BOOL AS
    $BODY$

    DECLARE

    BEGIN
        IF edit_point_distance(g1, g2) < e THEN
        RETURN TRUE;
      ELSE
        RETURN FALSE;
      END IF;

      RETURN FALSE;
    END

    $BODY$
    LANGUAGE 'plpgsql';

3) DROP FUNCTION IF EXISTS edit_point_distance( GEOMETRY, GEOMETRY );
CREATE OR REPLACE FUNCTION edit_point_distance(p1 GEOMETRY, p2 GEOMETRY)
  RETURNS FLOAT AS
$BODY$
```

```
DECLARE
   x1                FLOAT;
   x2                FLOAT;
   y1                FLOAT;
   y2                FLOAT;

BEGIN
   x1 = ST_X(p1);
   x2 = ST_X(p2);
   y1 = ST_Y(p1);
   y2 = ST_Y(p2);

   RETURN |/((x1 - x2)^2.0 + (y1 - y2)^2.0);

END

$BODY$
LANGUAGE 'plpgsql';
```

Algorithm 9: check the success of the trip segmentation

```
DROP FUNCTION IF EXISTS sim(id1 int,id2 int) CASCADE;
CREATE FUNCTION sim(id1 int,id2 int) RETURNS int[] AS $$
DECLARE
i tg_pair[];
j tg_pair[];
b float;
c float;
d int;
e double precision;
f double precision;
k int[];
x int;
m int;
n int;
p numeric;

    BEGIN
    SELECT ARRAY_AGG(trip_id) into k from trips where user_id=id2;
    SELECT tg_pair[] into i from trips where trip_id =id1;
    SELECT distance into e from trips where trip_id =id1;
    m=t_length(i);
    IF e+1!=0 THEN
    c=1000000;
```

```
FOREACH x in ARRAY k
LOOP
SELECT distance into f from trips where trip_id =x;
IF x!=id1 AND f+1!=0 THEN
RAISE NOTICE    'times here is %', x;
SELECT tg_pair[] into j from trips where trip_id =x;
n=t_length(j);
p=LEAST(m,n)/GREATEST(m,n)::numeric;
IF p>=0.7 THEN
b=t_edit_distance(i,j,0.001);
RAISE NOTICE    'edit distance is %', b;
IF b<c THEN
c=b;
d=x;
END IF;
END IF;
END IF;
END LOOP;
return ARRAY[d,c];
     ELSE
return ARRAY[0,0];
END if;
 END
 $$ LANGUAGE plpgsql;
```

Algorithm 10: tripleg segmentation evaluation

```
DROP FUNCTION IF EXISTS tripleg_evaluation(id1 int,id2 int) CASCADE;
CREATE FUNCTION tripleg_evaluation(id1 int,id2 int) RETURNS float AS $$
DECLARE
i int[];
j int[];
a text;
b text;
c1 int;
c2 int;
d double precision;
k int;
g1 geometry;
g2 geometry;
x int;
y int;

     BEGIN
     RAISE NOTICE    'id here is %', id1;
```

```
k=0;
SELECT number_of_triplegs into c1 from trips where trip_id =id1;
SELECT number_of_triplegs into c2 from trips where trip_id =id2;
IF c1=c2 THEN
SELECT trip_id into a from trips where trip_id =id1;
SELECT trip_id into b from trips where trip_id =id2;
   IF c1=1 THEN
   SELECT to_point_id into x from trips where trip_id=id1;
   SELECT to_point_id into y from trips where trip_id=id2;
   SELECT the_geom into g1 from locations where no=x;
   SELECT the_geom into g2 from locations where no=y;
  d=edit_point_distance(g1,g2);
   ELSE
   d=0;
SELECT array_sort(ARRAY_AGG(to_point_id)) into i from triplegs where
trip_id=a;
SELECT array_sort(ARRAY_AGG(to_point_id)) into j from triplegs where
trip_id=b;
LOOP
SELECT the_geom into g1 from locations where id_location=i[k+1];
SELECT the_geom into g2 from locations where id_location =j[k+1];
d=d+edit_point_distance(g1,g2);
k=k+1;
EXIT WHEN k=c1-1;
END LOOP;
      END IF;
return d;
ELSE
return -1;
END IF;
END
$$ LANGUAGE plpgsql;
```

Algorithm 11: check the success of the mode inference

```
DROP FUNCTION IF EXISTS sim_leg(id1 int,id2 int) CASCADE;
CREATE FUNCTION sim_leg(id1 int,id2 int) RETURNS int[] AS $$
DECLARE
i tg_pair[];
j tg_pair[];
b float;
c float;
d int;
e double precision;
f double precision;
```

```
k int[];
x int;
m int;
n int;
p numeric;

    BEGIN
    SELECT ARRAY_AGG(tripleg_id) into k from triplegs where user_id=id2;
    SELECT tg_pair[] into i from triplegs where tripleg_id =id1;
    SELECT distance into e from triplegs where tripleg_id =id1;
    m=t_length(i);
    IF e+1!=0 THEN
    c=1000000;
    FOREACH x in ARRAY k
    LOOP
    SELECT distance into f from triplegs where tripleg_id =x;
    IF x!=id1 AND f+1!=0 THEN
    RAISE NOTICE    'times here is %', x;
    SELECT tg_pair[] into j from triplegs where tripleg_id =x;
    n=t_length(j);
    p=LEAST(m,n)/GREATEST(m,n)::numeric;
    IF p>=0.7 THEN
    b=t_edit_distance(i,j,0.001);
    RAISE NOTICE    'edit distance is %', b;
    IF b<c THEN
    c=b;
    d=x;
    END IF;
    END IF;
    END IF;
    END LOOP;
    return ARRAY[d,c];
        ELSE
   return ARRAY[0,0];
   END if;
    END
    $$ LANGUAGE plpgsql;
```

Algorithm 12: mode inference evaluation

```
DROP FUNCTION IF EXISTS mode_evaluation(id int) CASCADE;
CREATE FUNCTION mode_evaluation(id int) RETURNS int AS $$
DECLARE
i int;
j int;
```

```
a int;
k int;

    BEGIN
    RAISE NOTICE    'id here is %', id;
        SELECT sim1[1] into i from triplegs where tripleg_id=id;
        SELECT transportation_type into j from triplegs where tripleg_id=id;
        SELECT transportation_type into k from triplegs where tripleg_id =i;
        IF j=k THEN
        a=1;
        ELSE
        a=0;
        END IF;
            return a;
        SELECT count(mode_eva) from triplegs where mode_eva=1;
    END
    $$ LANGUAGE plpgsql;
```