THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Preliminary Data Processing

2.1   Data Collecting and Cleaning

The taxi GPS data used in this project is collected from the Computational Sensing Lab[9] at Tsinghua University, Beijing, China. The data set contains approximately 83 million time-stamped taxi GPS records collected from 8,602 taxis in Beijing, from 1 May 2009 to 30 May 2009. The original data set consists of seven fields as shown in Table 2.1. The GPS coordinates in the data set are defined in the WGS-84[1] standard coordinate system, which is the reference coordinate system used by the GPS.

The original data set comes in a binary file format. After the data was decoded and imported into a MySQL database, the first step in cleaning data was to delete all records with zero value in the SPEED field, since when a taxi is stationary it yields no valuable information about the *trajectory* it is moving along. While being stationary could be due to a traffic jam, this kind of information is well captured by the time difference between the previous *non-stationary* data point and the next *non-stationary* data point.

---

[1]World Geodetic System

Field	Explanation
CUID	ID for each taxi
UNIX_EPOCH	Unix timestamp in milliseconds since 1 January 1970
GPS_LONG	Longitude encoded in WGS-84 multiplied by 10^5
GPS_LAT	Latitude encoded in WGS-84 multiplied by 10^5
HEAD	Heading direction in degrees with 0 denoting North
SPEED	Instantaneous speed in metres/second (m/s)
OCCUPIED	Binary indicator of whether the taxi is hired (1) or not (0)

Table 2.1: Fields in the original data set

Moreover, all records must have a *unique* pair of CUID and UNIX_EPOCH fields, since it is not possible for a taxi to appear in two different locations at the same moment in time. This is possibly due to some errors in aggregating the original data set.

2.2   Reverse Geocoding

After the data set was cleaned, the next step was to map each GPS data point to a road segment, which is also known as *reverse geocoding*. A number of algorithms[4] have been proposed for this purpose, but most of them require an additional GIS[2] database of the road network in Beijing. This project adopted an alternative strategy which leverages on the existing public API[3] for reverse geocoding.

Currently, a number of online mapping platforms provide reverse geocoding services as part of their developer APIs. Amongst others, Google Maps and Baidu Maps offer relatively stable and fast reverse geocoding services. However, due to the "China

---

[2]Geographic Information System
[3]Application Programming Interface

Field	Explanation
DataUnitID	Nominal primary key for each record
BD09_LONG	Longitude encoded in BD-09 format
BD09_LAT	Latitude encoded in BD-09 format
STREET	Street name

Table 2.2: Additional fields added to data set

GPS shift problem"[7] where coordinates encoded in WGS-84 standard are required by regulations to be shifted by a large and variable amount when displayed on a street map, Google Maps is not able to display a GPS data point correctly.

Baidu Maps, on the other hand, has been using their own coordinate system called BD-09 which is an improved version of the Chinese official coordinate system, GCJ-02. Baidu provides a set of APIs to convert WGS-84 coordinates into BD-09 ones. To store the converted coordinates as well as the street names obtained from reverse geocoding, four new fields were added to the original data set as shown in Table 2.2.

In order to use Baidu APIs for reverse geocoding, the following system architecture was set up as shown in Figure 2-1. The Apache HTTP server hides the MySQL database and sends HTTP POST request to Baidu Maps Web API to get converted coordinates. Then it updates the database through PHP *mysqli* utility.
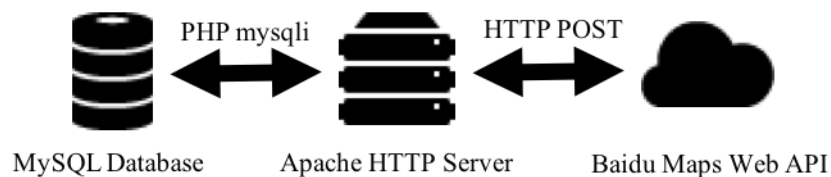

Figure 2-1: Basic system architecture

After the coordinates were converted from WGS-84 format to BD-09 format, Baidu

Maps Web API was used to reverse geocode all GPS data points. However, the system architecture was slightly changed, to accommodate the change in technology used. For reverse geocoding, AJAX[4] was used to communicate with the Baidu Maps Web API for speed and unlimited number of requests per day. Therefore, one additional layer was added to the existing system architecture as shown in Figure 2-2.
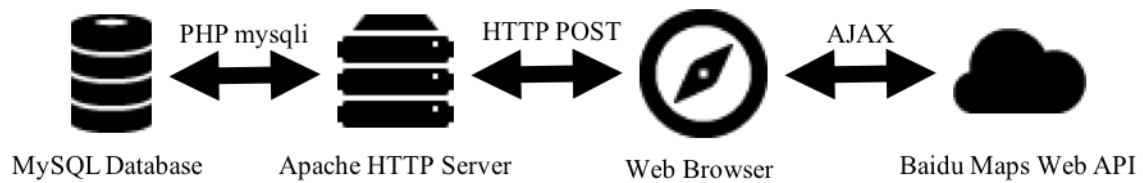

Figure 2-2: Augmented system architecture

Executed in a web browser environment, AJAX sent HTTP POST requests to the Apache HTTP server to fetch the converted coordinates in BD-09 format which were subsequently sent to the Baidu Maps Web API server *asynchronously* via HTTP GET requests. Once the server responded with the name of the road segment, AJAX updated the database by sending another HTTP POST request to the Apache HTTP server. The asynchronous nature of this architecture, however, caused a few problems which are addressed in Section 2.3.

2.3   Outlier Detecting

2.3.1   Motivation for Outlier Detection

The Baidu Maps Web API for reverse geocoding is stable and fast, but does not produce no errors. Sometimes, a GPS data point may be mapped to a main road but actually it is on the side road, which is one of the limitations mentioned in Section 1.2

---

[4]Asynchronous JavaScript and XML

or it is actually mapped to a street that Baidu Maps does not recognise. In neither case will Baidu Maps produce a correct reverse geocoding. Moreover, the reverse geocoding process is asynchronous, which means that it is being performed in the background in parallel with the main application thread. Therefore, it is inevitable that some street names may get lost when the records are being updated or a record is updated with a wrong street name. Figure 2-3 shows a drastic example.
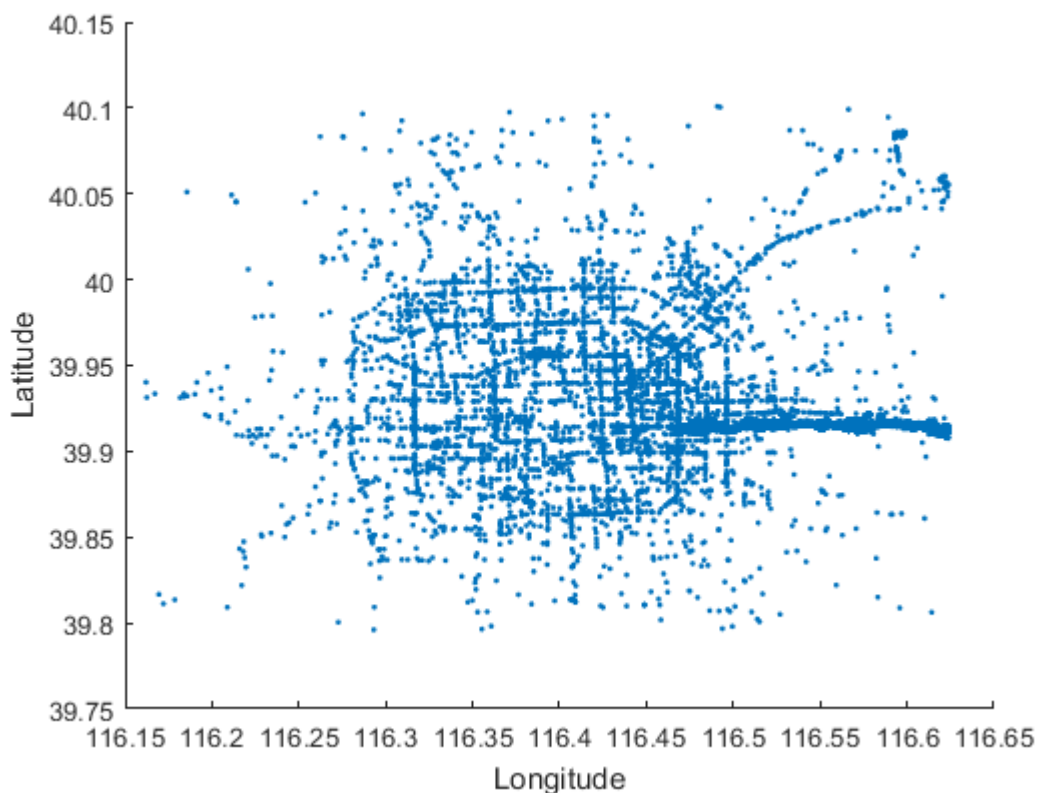

Figure 2-3: Example of outliers

In this example, the Baidu Maps believes that all data points plotted belong to a particular street. But when plotted on a 2-D plane, these data points almost represent the *entire* road network in Beijing. The actual, correct street is represented in the

figure as the *thickest* line on the right half of the figure with a longitude ranging from 116.45° to 116.65°. Erroneous records like those not on the thickest line are known as *outliers* and must be properly identified and removed. This project proposes a novel outlier detection approach based on unsupervised learning whose principle behind is based on Theorem 1.

Definition 3 (*Reasonable reverse geocoder*)**.** A reasonable reverse geocoder always gives its best matching from a GPS data point to a street whenever possible and has an accuracy more than 50%.

Theorem 1 (*Majority Clustering Theorem*)**.** If a *reasonable reverse geocoder* is used to reverse geocode a set of GPS data points which are mapped to a particular street *in reality*, then, when plotted on a 2-D plane, majority (more than 50%) of the points must be clustered together to form a rough shape that is similar to the shape of the street that they are supposed to be mapped to.

Proof. Proof by contradiction. Assume, for the purpose of contradiction, majority (more than 50%) of the data points that are *indeed* located on the same street are scattered arbitrarily on a 2-D plane after being reverse-geocoded by a reasonable reverse geocoder. In particular, when plotted on a 2-D plane, majority of them do not form a similar shape to that of the street they are supposed to be mapped to. Then, the majority must have been erroneously mapped to some other streets because no single street covers the whole city area. Thus, the reasonable reverse geocoder has only achieved an accuracy less 50%, which contradicts the Definition 3 of a reasonable reverse geocoder.   □

2.3.2   Outlier Identification

Apparently, Baidu Maps provides a reasonable reverse geocoder because it is of industrial-grade quality and has an accuracy larger than 50%. Therefore, if a set

of points belong to a particular street, after reverse-geocoded by Baidu Maps, majority of them should be clustered to assume a rough shape of that street according to Theorem 1. Based on that, an unsupervised learning technique — clustering can be used to separate the correctly mapped data points from outliers.

Many clustering techniques are available[5]. Since each record can be represented graphically by a point on a 2-D plane with BD09_LAT as the x axis and BD09_LONG as the y axis, a **self-organising feature map**[3] seems to be an appropriate technique to use.

A self-organising feature map is a type of artificial neural network. It consists of a pre-defined number of interconnected neurons randomly scattered among the data points at the begin of training. A neuron gradually moves to the centroid of the data cluster it represents as the training goes. Once the training stops, all data points near to a particular neuron, in terms of Euclidean distance[5], are assigned to the cluster that neuron represents.

Once the centroid of each cluster had been calculated, a distance threshold was set to identify and remove outliers so that whenever the minimum distance between a data point and all centroids is above the threshold, that data point will be considered as an outlier and removed. The python-like pseudocode in Listing 2.1 describes this idea.

For this project, two thresholds were set: 30 metres and 50 metres. This was to ensure that there was sufficient data for estimating travel time while the estimates were as least affected as possible by outliers. If the threshold is set to a too small value, the remaining data may not be sufficient for subsequent machine learning tasks; on the other hand, however, if the threshold is set to a too big value, the accuracy of the final results will be subject to the excessive outliers.

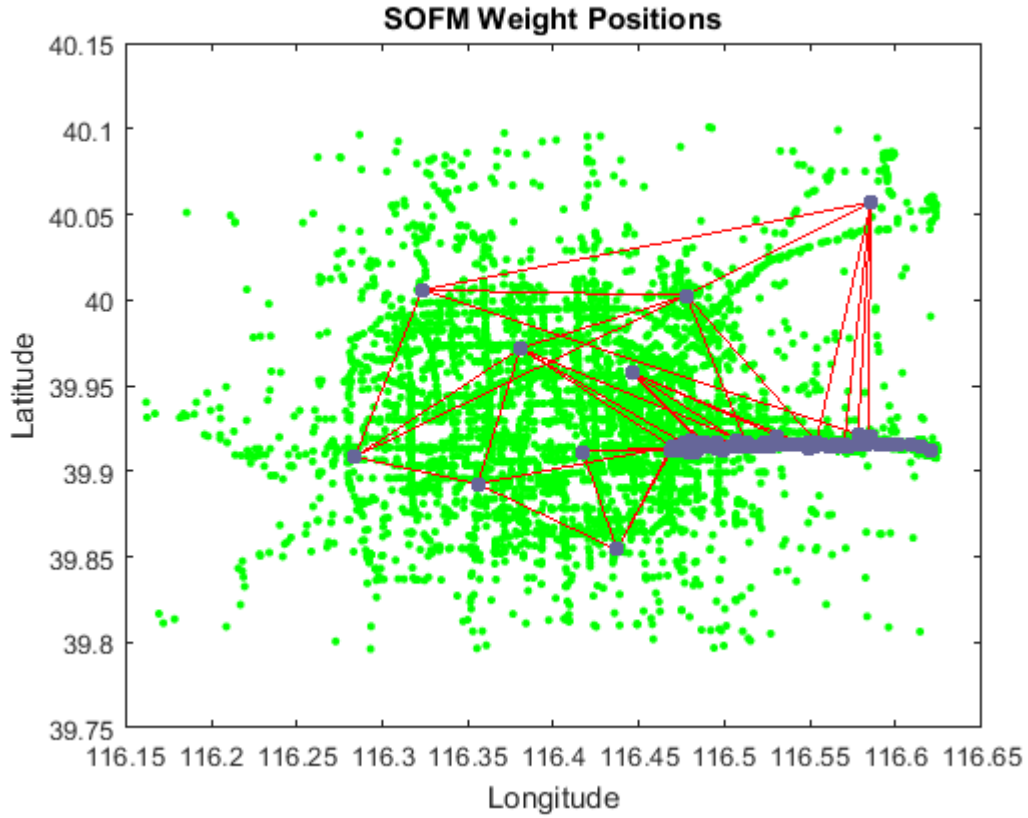[5]Other distance measures are also possible

Figure 2-4: Neuron positions after training

Listing 2.1: Pseudocode for outlier detection

```
for record in records:
        min_distance = math.inf
        for centroid in centroids:
                min_distance = min(min_distance, \
                        get_distance(record, centroid))
        if min_distance > threshold:
                remove(records, record)
```