# Anomalous Taxi Route Detection System Based on Cloud Services

Yu Zi[1], Yun Luo[2](✉), Zihao Guang[1], Lianyong Qi[3], Taoran Wu[4], and Xuyun Zhang[1]

[1] Department of Electrical, Computer and Software Engineering, University of Auckland, Auckland, New Zealand
{zgua779,zyu539}@aucklanduni.ac.nz, xuyun.zhang@auckland.ac.nz

[2] Faculty of Engineering and IT, University of Technology Sydney, Ultimo, Australia
Yun.Luo@student.uts.edu.au

[3] Qufu Normal University, Jining, China
liangyongqi@gmail.com

[4] Guizhou University of Finance and Economics, Guiyang, China
taoran.wu@mail.gufe.edu.cn

**Abstract.** Machine learning is very popular right now. We can apply the knowledge of machine learning to deal with some problems in our daily life. Taxi service provides a convenient way of transportation, especially for those who travel to an unfamiliar place. But there can be a risk that the passenger gets overcharged on the unnecessary mileages. To help the passenger to determine whether the taxi driver has made a detour, we propose a solution which is a cloud-based system and applies machine learning algorithms to detect anomaly taxi trajectory for the passenger. This paper briefly describes the research on several state-of-art detection methods. It also demonstrates the system architecture design in detail and gives the reader a big picture on what parts of the application have been implemented.

**Keywords:** Anomaly detection · Taxi route · Cloud service · Machine learning

## 1 Introduction

In the recent years, more and more organizations store, process, and extract value from data of all forms and sizes. With a certain amount of information available, it is inevitable that the big data will be continuously integrated into and influence our daily life. For example, researchers can use mobile phone data to analyze how people's location and traffic pattern influence the urban planning. This can help those urban planners to determine the best practices for stoplights, construction and parking. This is also a good example of applying machine learning methods. With the use of machine learning algorithm, we can learn patterns from given information and make predictions on data using the trained model.

In our daily life, taxi service plays a uniquely important role. It provides a convenient door-to-door way of transportation. That makes it very helpful to people who travel to an unfamiliar place. The taxi service usually charges passengers based on the time or mileages they have taken. However, taxi passengers can suffer from the risk of being overcharged on the unnecessary mileages incurred by taxi drivers intentionally or unintentionally. Due to the lack of background about the cities, most passengers cannot notice the subtle differences between the normal route and the altered one.

Smart phones equipped with GPS can be viewed as sensors which track the taxi trajectory. The advances in location-acquisition and mobile computing techniques has generated massive taxi trajectory data from GPS devices on taxis and mobile devices carried by the passengers. In this paper, we focus on constructing a system which can collect the trajectory data of the taxi trips, extract the temporal and spatial information, and analyse them using machine learning methods. This anomaly analysis can help the passengers know if the taxi has taken a normal route. It can also contribute to traffic management which is one of the most important aspects of smart cities. For example, if the travel distance of an anomalous trajectory is shorter than that of the normal routes, this route can be considered as one of the recommended routes for missioncritical drivers in emergency cases.

The rest of the paper is organised as follows. We review the related work in the next section. Section 3 states and analyse the targeted problem and our goal. We describe the system overview in Sect. 4 and the details of implementing the system in Sect. 5, including development tools and frameworks involved, data preprocessing, and anomalous taxi trajectory detection algorithm. In Sect. 6, we conduct a suite of experiments on real-life data to evaluate the proposed system. Finally, Sect. 7 concludes our research work and points out the future work.

## 2 Related Work

### 2.1 Existing Solutions

We tried to find out mobile apps with the functionality of anomaly route detection, from the app store. Unfortunately, by now there does not exist such an app in the market that provides the functionality we concern.

A map application like Google Maps can predict the directions to the destination for the user. The passenger may compare this predicted route with the taxi route tracked by the GPS on the mobile device. However, the routes recommended by the map application are quite limited. For example, even the route taken by the taxi driver is not shown as one of the recommended directions provided by Google Maps, this route may still be normal due to current traffic condition. Another case is that if the road network on the digital map is not up-to-date, the directions provided by the map application cannot be considered as accurate. Hence this method is not strong enough for the passenger to determine if the taxi has taken an anomaly route or not.

### 2.2 Trajectory Outlier Detection Methods

Some researchers have published papers which propose methods for anomalous trajectories detection. iBoat is a real-time detection method and can also identify which parts of the trajectory are responsible for its anomalousness [3]. The project team claims that the method has an excellent accuracy and overcomes many shortcomings of other state-of-the-art methods. iBoat compares a test trajectory against a set of sampled historical trajectories with the same SourceDestination pair, rather than using time and distance to directly judge whether it is anomalous or not. We also found another paper on anomaly taxi trajectory detection. The method is named as iBAT [10]. iBAT exploits anomalous trajectories intrinsic properties of being few and different, and applied the isolation mechanism to detect anomalous trajectories. Another research group developed a taxi driving fraud detection system [4]. They mainly considered two aspects of evidence:travel route evidence and driving distance evidence. Based on the Dempster-Shafer theory, those two aspects of evidence are combined to perform the detection. Besides, on Fisher's paper [9], they demonstrated a novel human assisted learning/classification framework for identifying anomalous behaviour on the basis of motion trajectories.

## 3 Problem Statement

Taxi passengers can suffer from the risk of being overcharged on unnecessary detours. To ensure the quality of taxi services, it is crucial to detect such cases and penalize the corresponding driver. Currently, the detection process is often done by experienced staff via manually checking the geolocation trajectory of a taxi trip. But this is costly and not effective because sometimes the passengers cannot even notice the subtle differences between a normal route and the altered one. And also the accuracy of manual detection is not quite reliable.

In this paper, we aim to develop an anomalous taxi route detection system that helps the end user to know if the route the taxi driver has taken is normal or not. The anomalous trajectories detection process should be done automatically rather than manually. Thus, we need to apply a smart state-of-the-art machine learning methods to achieve this purpose. In addition, the system should be deployed on the cloud so that it can be easily scaled on demand. As the potential end users of the system are the taxi passengers, a mobile application needs to be developed as the interface between the users and the system, so that the users can access the detection system remotely. As well, the mobile devices are responsible for collecting geolocation points of each taxi ride.

## 4 System Overview

To achieve the goal mentioned above, we need to implement a client-server architecture which consists of a front end mobile application and a back end server. The mobile application should provide functionality of recording and storing the location information during the taxi trip. A tunnel needs to be built up as well so that the client and the server can communicate with each other. Mover, a anomaly detection algorithm for trajectory data should be implemented using machine learning methods. This algorithm will be integrated into the architecture and can return meaningful results with a given route dataset. The back end server will be deployed onto the cloud to achieve high scalability.

The system overview of our solution is shown in Fig. 1. The back-end server, MapReduce data processing module and the database are deployed on the cloud. To detect anomalous routes, we utilize a machine learning method. Thus a huge amount of raw taxi traffic data is required as the training dataset. As the original raw data does not meet the requirement for the detection algorithm, the raw data firstly gets pre-processed by MapReduce module and then stored in the database. The mobile application is responsible for route data collecting and acts as the bridge connecting the end-users and the back-end detection system. Once the taxi arrives at the destination, the application will send the recorded trajectories data to the back-end servers. The anomalous detection algorithm runs on the server will then take samples from database and use them to analyze the received test trajectory. At last, the resulted score will be returned.

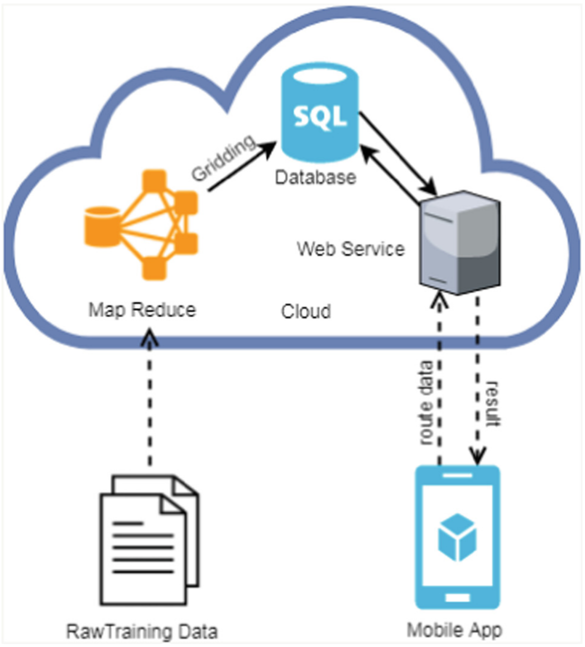


**Fig. 1.** System overview.

In general, the application is developed as a client-server architecture as shown in Fig. 2. As MapReduce data processing module only relates to the functionality of raw training data processing, it is not considered as one of the primary modules hence we omit it when we discuss the architecture.

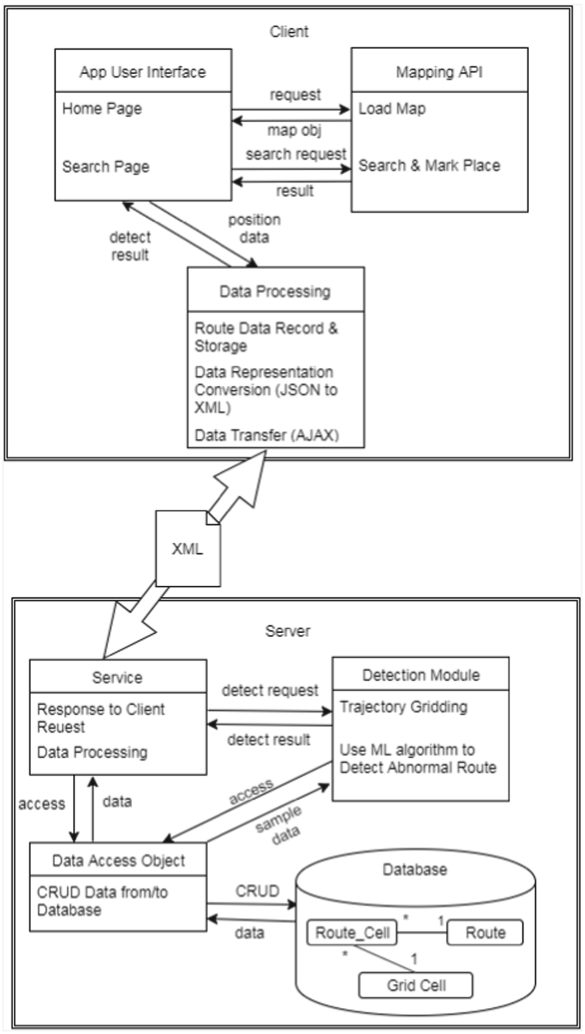


**Fig. 2.** Architecture diagram.

The client side is split into three modules. App User Interface (UI) module consists of two pages whose structures and styles are described by corresponding HTML and CSS files. The Mapping API module should provide functionality related to mapping service. It passes a map object to the App UI module when the application loads the home page for the first time. This module can also predict the location based on the text entered by user, and return the geographical location back to the UI. Data Processing module is the core logic part of the front end. It is responsible for route data recording, data transforming and interacting with the back end side. The current position data is obtained from the UI module using HTML Geolocation API and stored in Data Processing module. Once the user reaches the destination, Data Processing module stops recording the current position, and then transform the dataset into XML form before sending it to the server. When the detected result is received, it will be passed to the UI and shown to the user.

The server side is split into three modules as shown in the diagram. Service module can deal with the request from the client and reformat the received raw data. Anomalous route detection methods and the trajectory gridding algorithm are packed into the Detection Module. It can process the data by map gridding and then analyze the given test trajectory. The map gridded taxi trajectory will be stored in the database as well. Every time a module tries to access the database, a data access object will be used to perform create, read, update and delete (CRUD) operations.

## 5 System Implementation

### 5.1 Implementation Environment

We develop the server side of the app using Java Enterprise Edition (Java EE) [8]. Java EE is the industry standard for developing portable, robust, scalable and secure server-side Java applications. To support the web application development, it provides web services, component model, management, and communications APIs. Thanks to the flexibility of Java, our solution can be deployed across platforms. We implement web services on back end side following REST style [7]. REST refers to Representational State Transfer which is an architectural style that specifies constraints, such as the uniform interface. It promotes scalability, modifiability and good performance. In our implementation, HTTP is used as the protocol and it provides API for the web service. The processed taxi trajectory data for machine learning algorithms is stored in a MySQL database [6].

The mobile application is developed using HTML, CSS and TypeScript, as we build the application with the hybrid mobile application approach. The mobile application of our solution is designed to be run on not only a specific mobile operation systems as the popular mobile OS like iOS and Android have certain portions of market share. A good way is to develop mobile apps for different mobile platform, which is a hybrid application. It is basically a web application which is developed using HTML, CSS and JavaScript, and then wrapped in a native application using platforms like Cordova, also known as write once and run everywhere approach. Ionic Framework [1] which is an open source project with a licence under MIT is used for front end development. It provides over 120 native plugins to utilize the native device features.

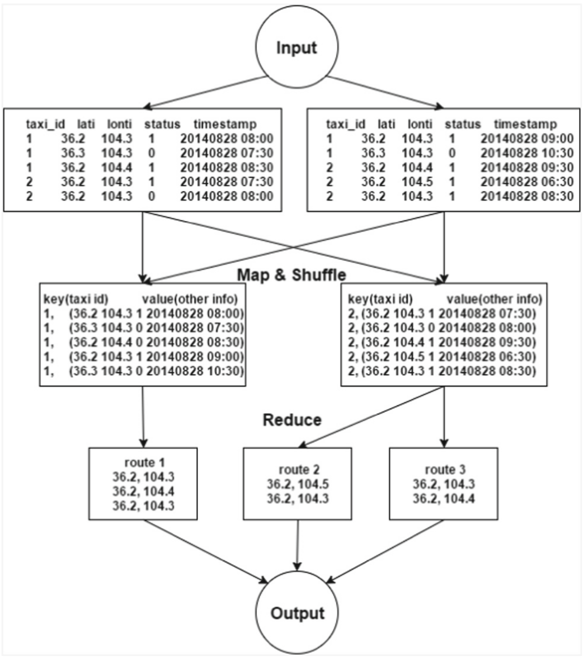
On the front end side, Google Maps JavaScript API [5] are integrated into the mobile application. With the use of mapping API, the trajectory data can be visualized by displaying location points on a map so that the user knows the route the taxi goes through. Google Maps API provides auto-complete predictions service which allows the text box to retrieve auto-complete results based on the user input for the trip destination. It provides place service as well which can retrieve the precise geographical location information of the destination set by the user. It also meets all requirements related to map visualization of our solution.

Raw taxi traffic data is required to be pre-processed before getting stored in the database. However, due to the computing speed and memory size issues, it is very time-consuming to process the data if the size of the data expands to a certain amount. To overcome this computational bottleneck, We implement the data pre-processing module with the use of Hadoop MapReduce [2]. MapReduce is a framework that allows programmers write applications to process a huge amount of data on large clusters of commodity hardware in parallel. It mainly contains two tasks called Map and Reduce. Map task takes a set of data as input, and converts it into another set of data. As the result, the individual elements are broken down into tuples of key-value pairs. Reduce task takes the output from a Map as an input and then combines those tuples into a smaller set of tuples. The major advantage of MapReduce is the scalability. It is easy to scale the data processing over multiple computing nodes. Once the data-processing module is implemented in the MapReduce form, we can just make a configuration change to scale the module to run on over hundreds, or even thousands of machines in a cluster. Besides, we do not need to care about how the data-passing works during a MapReduce job, as the framework can manage all the details of data-passing around the cluster. The detail of MapReduce version of data pre-processing module will be discussed subsequently.

### 5.2 Data Pre-processing

1) Trajectory Sorting and Splitting: A taxi trip trajectory record consists of taxi id, a geolocation point represented by latitude-longitude pairs, generated by the GPS device on that taxi, time stamp and the service status associated with that time stamp. A complete taxi trajectory can be obtained by connecting those geolocation points in the order of time stamp. As the taxi overcharging problems only occur when the taxi is in service, i.e. carrying the passenger, we need to split those taxi trajectories corresponding to the taxi trips from the raw dataset. However, the order of the raw taxi trajectory dataset cannot be guaranteed. That means the dataset needs to be sorted before splitting. Since the anomalous trajectory detection algorithm utilizes a machine learning method, a huge amount of taxi trajectories data are needed for the better performance of detection. During the earlier development of data pre-processing module, we encountered an issue related to memory bottleneck, as the whole dataset needs to be loaded into memory in order to sort the trajectory points for a taxi. Instead, in the current solution, Hadoop MapReduce is used to achieve the trajectory sorting and splitting tasks as mentioned in the Decision Making on Tool Selection sub-section.

As shown in Fig. 2, at the first step, the raw taxi trajectory dataset is fed into mappers, which will form keyvalue pairs from single geolocation points. In our case, the taxi id is defined as key and all other information is defined as value so that the whole dataset can be effectively separated into small pieces. Then the shuffler will group pairs with the same key (i.e. same taxi id) together and send them to the reducer. The next step is to extract taxi routes from the set with specific taxi id. As mappers and reducers do not interfere with each other as shown in Fig. 3, the map and reduce tasks could be parallelized easily. That means it will take much shorter time to extract routes from the raw dataset than using the earlier implementation theoretically.



| taxi_id | lati | lonti | status | timestamp |
|---|---|---|---|---|
| 1 | 36.2 | 104.3 | 1 | 20140828 08:00 |
| 1 | 36.3 | 104.3 | 0 | 20140828 07:30 |
| 1 | 36.2 | 104.4 | 1 | 20140828 08:30 |
| 2 | 36.2 | 104.3 | 1 | 20140828 07:30 |
| 2 | 36.2 | 104.3 | 0 | 20140828 08:00 |

| taxi_id | lati | lonti | status | timestamp |
|---|---|---|---|---|
| 1 | 36.2 | 104.3 | 1 | 20140828 09:00 |
| 1 | 36.3 | 104.3 | 0 | 20140828 10:30 |
| 2 | 36.2 | 104.4 | 1 | 20140828 09:30 |
| 2 | 36.2 | 104.5 | 1 | 20140828 06:30 |
| 2 | 36.2 | 104.3 | 1 | 20140828 08:30 |

| key(taxi id) | value(other info) |
|---|---|
| 1, | (36.2 104.3 1 20140828 08:00) |
| 1, | (36.3 104.3 0 20140828 07:30) |
| 1, | (36.2 104.4 0 20140828 08:30) |
| 1, | (36.2 104.3 1 20140828 09:00) |
| 1, | (36.3 104.3 0 20140828 10:30) |

| key(taxi id) | value(other info) |
|---|---|
| 2, | (36.2 104.3 1 20140828 07:30) |
| 2, | (36.2 104.3 0 20140828 08:00) |
| 2, | (36.2 104.4 1 20140828 09:30) |
| 2, | (36.2 104.5 1 20140828 06:30) |
| 2, | (36.2 104.3 1 20140828 08:30) |

route 1
36.2, 104.3
36.2, 104.4
36.2, 104.3

route 2
36.2, 104.5
36.2, 104.3

route 3
36.2, 104.3
36.2, 104.4

**Fig. 3.** The work flow of MapReduce program.

2) Map Gridding and Trajectory Augmenting: As a preparation of anomalous trajectory analysis, the city map is split into grid-cell with equal size. Then each taxi trajectory is mapped to grid cells and can be represented by a sequence of traversed cells. However, the GPS devices on taxi record a geolocation at a low frequency in practice. Therefore a map-gridded taxi trajectory may consist of some sequences of geolocation points which are not adjacent to each other, i.e. there exists a gap between some of the GPS points. Thus, we need to fill up those gaps by inserting pseudo cells to ensure that the same taxi trajectories can be represented equally in the system. This operation is called trajectory augmenting. Once the route information is extracted from

the raw dataset and gets map-gridded, it can then be stored in the back-end database.

During the map gridding process, the size of the grid cell is an important parameter we need to care about. It affects the performance of the anomalous trajectory detection algorithm. If the size is too small, the city map will be separated into a huge number of grid cells. This results in a sparse sample set when sampling from the training data. This makes it difficult to perform evaluation and testing. Also, routes with similar shapes and location points may be considered as different which is not what we expect. If the size is too large, the sequence of gridded points cannot reflect the shape and trend information of the original trajectory.

## 5.3 Anomalous Trajectories Detection Algorithm

In our research work, we address the problem of anomalous trajectory detection by using iBAT (isolation based anomalous trajectories detection) method which is described in [10]. Instead of comparing trajectories based on distance or density measure, iBAT method detects anomalous route based on the following properties of anomalous trajectories:

- anomalous trajectories are few in number;
- they are different from the majority, in particular, they pass different locations, or pass similar locations in different orders.

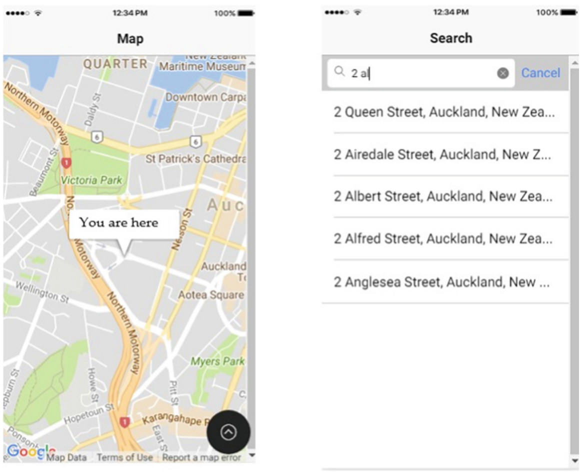
Here we briefly describe the workflow of iBAT method. It is a lazy learning algorithm, which does not train a model until a test sample is given. iBAT method tries to separate the given test trajectory from the rest of trajectories with the same source and destination, by randomly picking cells solely from the test trajectory. For example, if $t$ is a test route and the rest form the sample set, we randomly select one cell from $t$ and remove the trajectories that do not pass the selected cell from the sample set. This process is repeated until no trajectory is left or all the trajectories left contain all the cell $t$ has. Since anomalous trajectories are few and different, most grids contained by anomalous trajectories are not contained by normal ones. Therefore, the anomalous routes can be easier to be isolated, i.e. the expected number of used grid cells to isolate an anomalous route should be much smaller.

## 5.4 Back End Web Service

On back end side, a runnable web service is implemented. It provides an API so that the front end client can get the time interval parameter from the server and send the route dataset for a taxi ride to the server. Once received the raw route data, this service re-formats the trajectory first. The Detection module performs map gridding and trajectory augmenting on this test trajectory and then retrieves those routes with the same source and destination grid as the test trajectory, from the database. And these route information will then be used to analyze the test trajectory by iBAT method. Currently, the back end web service is deployed and being tested on the cloud computing platform provided by Unitec Institute of Technology.

## 5.5 Mobile Application

On front end side, the mobile application has been implemented. The application basically has a simple user interface with two pages, map page and search page as shown in Fig. 4. Here we briefly describe the work flow of the mobile application. When the mobile application is launched, the map page will show up, and mark the user's current location as shown on the left screen in Fig. 4. To set up the taxi trip destination, the user needs to navigate to the search page by pressing the round floating button at the bottom right corner first then pressing the pop-up button with a search icon to navigate to the search page. Initially, the search page only shows a header bar and a text box which allows the user to enter the address of the destination. A result list will show up and display predicted locations based on the text entered. Once the user pressed one of the location, it will navigate back to the home page. Then a marker is pinned at the centre of the map showing the selected destination on the map. If the user is satisfied with the set destination and wants to start to record the taxi ride trajectory, the user needs to press the start button to trigger the geolocation record logic. Before reaching the destination, the route points are recorded at a pre-set interval which is 2 s by default. The system manager can change this interval by setting the corresponding parameter at the back-end server.



2 al

2 Queen Street, Auckland, New Zea...

2 Airedale Street, Auckland, New Z...

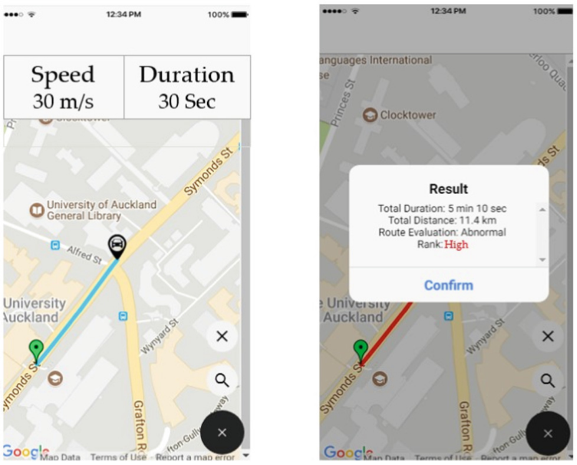
2 Albert Street, Auckland, New Zea...

2 Alfred Street, Auckland, New Zea...

2 Anglesea Street, Auckland, New ...

**Fig. 4.** Mobile application screen shots-initial screen and search page.

Meanwhile, the map page displays the route the taxi has passed, the current speed of the taxi and the accumulated duration of the taxi ride as shown in Fig. 5. Once the taxi gets close enough to the pre-set destination point, a dialog will pop up to inform the user. The user can then confirm to stop the record logic or press "keep going" button to keep record the taxi trajectory until the user presses the stop button. Once record logic is stopped, the complete trajectory will be sent to the back-end server for anomaly analysis. The result of the detection will be returned in the form of a score scaled from 0 to 1. The higher the score, the more abnormal the taxi trajectory. The route will be marked by a specific color based on the resulted score. As shown in Fig. 5, we assume that the resulted score is at a high rank, thus the route and the status are marked as red.


Speed
30 m/s
Duration
30 Sec

Result
Total Duration: 5 min 10 sec
Total Distance: 11.4 km
Route Evaluation: Abnormal
Rank: High
Confirm

**Fig. 5.** Mobile application screen shots-status screen and result screen. (Color figure online)

### 5.6 Data Transfer Between Client and Server

The mobile application can communicate with the back end server using HTTP protocol. When the recording is triggered, the mobile app firstly sends a request with GET to the server to acquire the number of record interval. Once the recording is stopped, the mobile app sends a POST request to the server with the location information dataset in XML form. As respond, the resulted anomalous score will then be returned to the front end mobile app.

## 6 Evaluation

### 6.1 Experimental Setup

For evaluation purpose, we use a real-world taxi geolocation dataset, which is collected from taxis served in Chengdu, China for about half a month. The total number of records is over 1.4 billion. Each record consists of the taxi id, service status, geolocation point and the corresponding time stamp. This is consistent with the format requirement for data pre-processing. For the sake of simplicity in computation and visualization, we restrict our interest within the city center of Chengdu with longitude [103.9E, 104.2E] and latitude [30.5N, 30.8N]. The size of each grid cell is set to be 250 m $\times$ 250 m and the map is split into 75 $\times$ 75 grid cells. To ensure the accuracy and stability of the anomalous route detection algorithm, two parameters (the number of trial m and sub-sample size) need to be adjusted properly. We use m = 40 and s = 250 in our experiments which is quite reasonable based on a 10-fold evaluation.

## 6.2 Trajectory Visualization

With the assist of visualization, we can compare the sample trajectories with the corresponding anomalous score to check if the resulted score is reasonable. To visualize a gridded trajectory, we calculate the geolocation of the center of each grid cell and connect the geolocation points in a proper order to reshape the trajectory. Then the processed trajectories set will be transformed into a file in GPX format in order to be displayed on Google Maps. During the experiment, we found that there are some cases where a part of the trajectory keeps looping in a region consists of a few grid cells. This can be caused by the unstable GPS signal during the taxi ride. Thus, we mesh such geolocation point sequences before visualizing the trajectories.

Here we show an example of trajectory visualization on a relatively small dataset. As shown in Fig. 6, it is quite obvious that route A and route B behave differently from the others. We checked the corresponding results and found that the anomalous score of route A is 0.874 and route B is 0.732, while the scores for 85% of the rest trajectories are below 0.5. We can say that route A and B can be successfully marked out as anomalous routes by the implemented detection algorithm. But we don not have enough evidence to say that all of the anomalous routes can be successfully classified as abnormal.

## 6.3 Anomalous Statistics

We conducted another 5 groups of experiments on the Chengdu taxi trajectory dataset. Specifically, we tried 5 differents pairs of origin and destination, labelled from A to E. Here we show five of the visualization results from Figs. 7, 8, 9, 10 and 11. Since it is impossible for us to mark all the anomalous routes and compare with the corresponding score, we only calculate the ratio of anomalous trajectories based on the resulted scores. The corresponding ratio is shown in Table 1.

## 6.4 Abnormality Rank

After visualizing the sampled trajectories, following are the findings determining the abnormality rank of trajectories based on our observation:

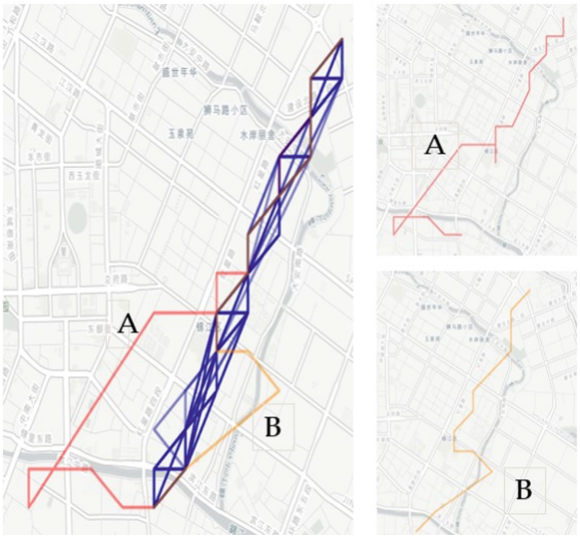


**Fig. 6.** An example of trajectory visualization on a relatively small dataset.

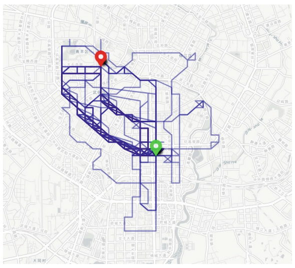


**Fig. 7.** Trajectory visualization A.

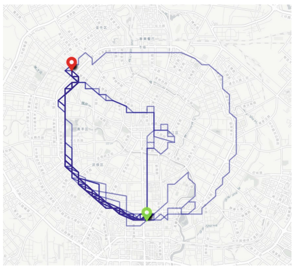


**Fig. 8.** Trajectory visualization B.

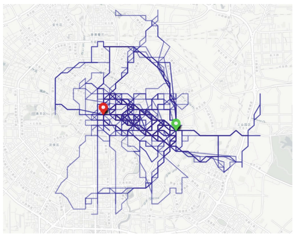


**Fig. 9.** Trajectory visualization C.

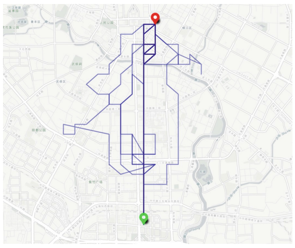


**Fig. 10.** Trajectory visualization D.

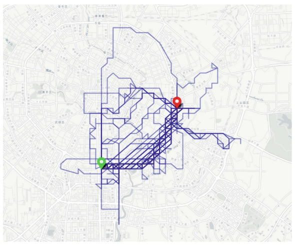


**Fig. 11.** Trajectory visualization E.

**Table 1.** Anomalous statistics.

| | Routes | Anomalous | Ratio |
|---|---|---|---|
| A | 567 | 42 | 7.4% |
| B | 327 | 43 | 13.15% |
| C | 1596 | 73 | 4.57% |
| D | 150 | 18 | 12% |
| E | 631 | 68 | 10.8% |

- 0–0.5: normal. This means many taxi drivers take the similar route.
- 0.5–0.7: possibly anomalous. They are usually similar to normal routes in most segments while only a few are different.
- 0.7–1: anomalous. They have few identical segments with the normal.

## 7 Conclusions and Future Work

In this paper, we have implemented an anomaly route detection system which can help users, the passengers who take a taxi, to know if the taxi has taken an anomaly route or not. The core part of the system have been implemented based on an anomalous route detection algorithm named iBAT. The backend server has been deployed and validated on a cloud computing platform. The implemented detection algorithm has been evaluated on a real-life taxi trajectory dataset collected from Chengdu, China. In the future, we will consider working on real time data transfer and detection and implementing the MapReduce version of anomalous route detection algorithm.

**Acknowledgement.** This work was supported in part by the New Zealand Marsden Fund under Grant No. 17-UOA-248, and the UoA FRDF fund under Grant No. 3714668.

## References

1. Build amazing native apps and progressive web apps with ionic
2. Welcome to apache hadoop. https://hadoop.apache.org/old/
3. Chen, C., et al.: iBOAT: isolation-based online anomalous trajectory detection. IEEE Trans. Intell. Transp. Syst. **14**(2), 806–818 (2013)
4. Ge, Y., Xiong, H., Liu, C., Zhou, Z.H.: A taxi driving fraud detection system. In: IEEE 11th International Conference on Data Mining, pp. 181–190 (2011)
5. Google: [9] google maps APIs - google developers. https://developers.google.com/maps/documentation/
6. MySQL: Why MySQL? https://www.mysql.com/why-mysql/
7. Oracle: The Java EE 6 tutorial. https://docs.oracle.com/javaee/6/tutorial/doc/
8. Oracle: Java EE at a glance. https://www.oracle.com/technetwork/java/javaee/overview/javaee-135128.html

9. Sillito, R.R., Fisher, R.B.: Semi-supervised learning for anomalous trajectory detection. In: BMVC, vol. 1, pp. 035–1 (2008)
10. Zhang, D., Li, N., Zhou, Z.H., Chen, C., Sun, L., Li, S.: iBAT: detecting anomalous taxi trajectories from GPS traces. In: ACM 13th International Conference on Ubiquitous Computing, pp. 99–108 (2011)