AI Communications 31 (2018) 33–46
DOI 10.3233/AIC-170747
IOS Press

An intelligent system for taxi service: Analysis, prediction and visualization

Yu Lu [a,b], Zeng Zeng [b], Huayu Wu [c], Gim Guan Chua [b] and Jingjing Zhang [d,*]
[a] *Advanced Innovation Center for Future Education, Beijing Normal University, Beijing, China*
[b] *Institute for Infocomm Research (I2R), A*STAR, Singapore*
[c] *Nanyang Technological University, Singapore*
[d] *Big Data Centre for Technology-Mediated Education, Beijing Normal University, Beijing, China*

Abstract. The fast advancements in sensor data acquisition and vehicle telematics facilitate data collection from taxis and thus, enable building a system to monitor and analyze the citywide taxi service. In this paper, we present a novel and practical system for taxi service analytics and visualization. By utilizing both real time and historical taxi data, the system conducts the estimation on region based passenger wait time for taxi, where recurrent neural network (RNN) and deep learning algorithms are used to build a predictive model. The built RNN-based predictive model achieves 73.3% overall accuracy, which is significantly higher than other classic models. Meanwhile, the system conducts the analytics on the taxi pickup hotspots and trip distributions. The experimental results show that around 97% trips are accurately identified and more than 200 hotspots in the city are successfully detected. Moreover, a novel three dimensional (3D) visualization together with the informative user interface is designed and implemented to ease the information access, and to help system users to understand the characteristics and gain insights of the taxi service.

Keywords: Recurrent neural network, deep learning, passenger wait time, intelligent transportation system, urban computing

1. Introduction

In densely populated large cities, especially in Asia, taxis are pervasively used in people's daily life, such as for individual travels between home and office during working days, family travels for shopping and dining during weekends, or foreign tourists visiting local attractions, etc. The pervasive taxi usage easily results in the spatiotemporal imbalance of taxi supply and demand, as well as the complexity of taxi operation patterns. Proper estimations on such imbalance and complex patterns would not only benefit the local taxi operators and passengers, but also help relevant government agencies to improve taxi service quality and eventually increase the productivity of citywide taxi service.

On the other hand, the abundance of taxi information, such as the taxi's real time GPS locations and operation status, becomes available and can be collected through the in-vehicle telematics system. For example, all the taxis in Singapore periodically update their locations, status (e.g., FREE or ONCALL, etc.) and other operation related information to the backend system. Such taxi data would directly help to extract the key information for desired analytics systems.

A number of previous studies attempt to utilize taxi information to address the critical analytics issues, typically including the prediction of taxi demand at taxi stand [26], recommendation of next-passenger for taxi driver [19], and recommendation of empty-taxi for passenger [28]. However, the previous studies often simply quantify the taxi demand using the historical pickups, while the actual taxi demand in reality is often constrained by the corresponding supply. Moreover, the previous studies seldom investigate and model the relationships and imbalance between taxi supply and demand. In addition, there is a lack of a well designed visualization and user interface to enable the system users to quickly understand the taxi analytics results and easily gain the hidden insights.

In this work, we endeavour to tackle and address the above issues, by considering the dynamic relationships between taxi supply and demand, and meanwhile leverage on the latest machine learning techniques,

*Corresponding author. E-mail: victoryluyu@gmail.com.

namely RNN with deep learning algorithms. We summarize the key contributions of this work as follows:

– We propose and implement a novel and practical system, utilizing both of the historical and real time taxi data, to conduct the analytics on the key perspectives of the taxi service, including the wait time estimation for the passengers, taxi hotspot detection, and trip extraction.
– We derive novel and effective features from the large scale taxi data, and use such features to build predictive model. The model is mainly used to estimate the region based passenger wait time, where the recurrent neural network (RNN) model and GPU based deep learning computational framework are investigated and applied successfully.
– We design a novel 3D visualization and informative user interface to help to access and understand the analytics results.

The rest of this paper is organized as follows: the related work is given in Section 2, and Section 3 presents the overall system architecture as well as the data collection. Sections 4, 5 and 6 depict the four modules in details respectively. System evaluation is conducted in Section 7, and we conclude our work in Section 8.

2. Related work

Driven by the availability of abundant information from taxis, taxi trace analytics has received massive attentions from both academia and industry in recent years. The relevant work can be generally classified into three categories: 1) mining taxi traces to study the city population movement patterns and behaviors [23,26]; 2) using taxi traces as a probe to infer or predict traffic conditions for city road networks [3,20]; 3) mining taxi traces to discover and sense human or vehicle's special events and behaviors [24,37]. For example, the authors in [37] utilize taxi traces to sense vehicle refueling behavior and citywide consumption. We refer the interested readers to a good survey [6] for taxi trace analytics.

RNN is one branch of neural networks and a powerful model for sequential data. Initially the vanishing gradient problem [4] and unmatched computational power limited its capabilities on solving real-life applications. In recent years, the long short-term memory (LSTM) [15] units partially solve the vanishing gradient problem and the GPU based computing architectures [31] significantly reduce training time of RNN. RNN has been widely used in a variety of tasks to support sequential data, including speech recognition [30], handwriting recognition [9], machine translation [21], etc. The deep RNNs, which combines multiple levels of representation, have proved so effective in deep networks with the flexible use of long range context. Alexandre et al. [8] adopt RNN, which consists of two hidden layers, to make predictions on taxi destinations based on the beginning of the taxi trajectory.

For the hotspot detection, the density-based clustering has been applied by analyzing the vehicle trajectories [17,29], and a variety of the existing algorithms can be used, such as CLARANS [27], CURE [14] and DBSCAN [13]. In this work, we also adopt the density-based clustering algorithm (i.e., DBSCAN) to conduct the hotspot detection task, but combine with the taxi trip extraction results to dynamically provide both the pickup and dropoff hotspots.

Visualization is an important tool to help people understand the data and the analytics results, especially for the large amounts of moving and trajectory data [38]. Adrienko et al. [1] proposes a framework for analysis by combining interactive visual displays with database operations and computational methods, and it effectively supports human perception, cognition, and reasoning. The animated maps [10], interactive cubes [16] as well as the aggregation-based techniques such as temporal histogram [11], are widely used to visualize moving and trajectory data. We adopt the similar design philosophy but innovate in using the 3D visualization techniques to describe our taxi analytics results in a dynamic way, especially for the extracted taxi trips and the predicted passenger wait time.

To the best of our knowledge, it is the first work applying RNN-based model to conduct the taxi passenger wait time estimation, although another RNN-based model has been applied to solve the taxi destination predication problem [8]. Moreover, a specifically designed 3D visualization solution with an informative user interface is designed and implemented, which effectively help users to access the key taxi analytics results, including wait time, trips and hotspots.

3. System design

3.1. System overview

The proposed system targets on conducting the analytics on the key aspects of taxi service and meanwhile

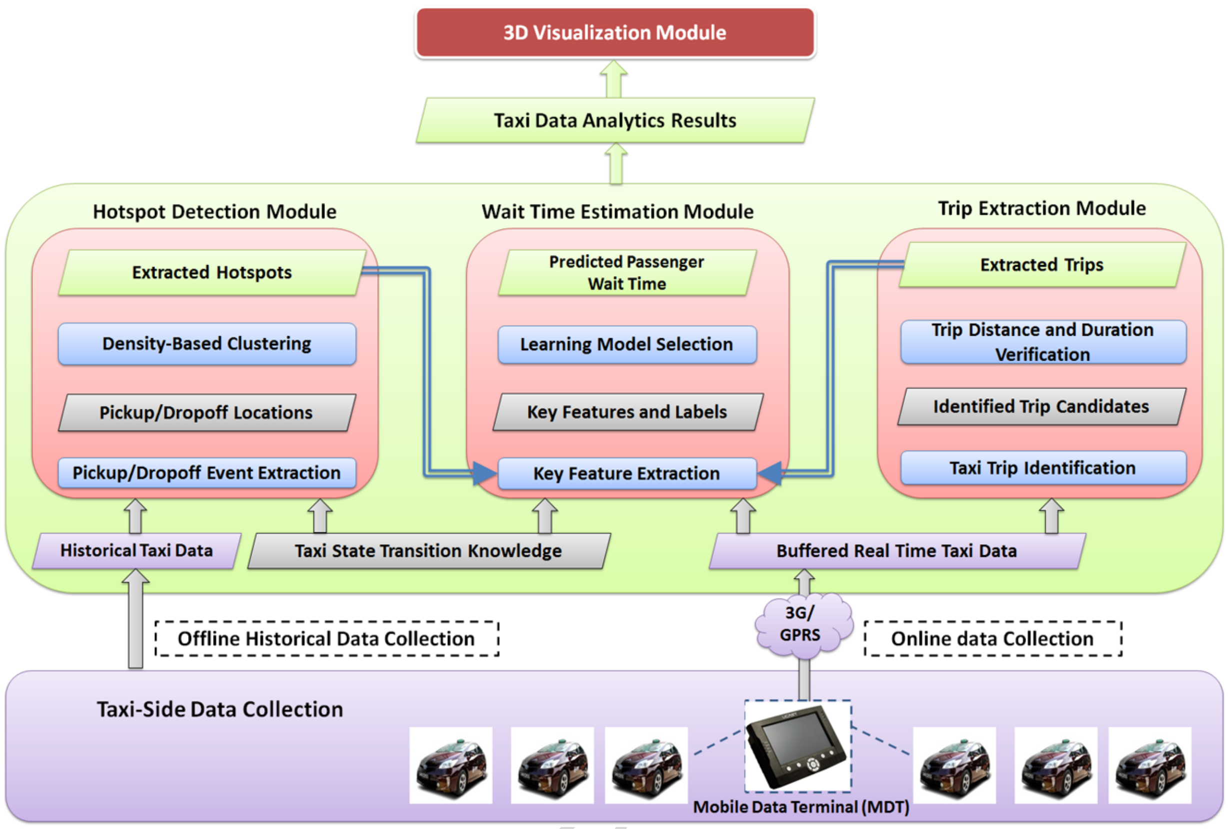
3D Visualization Module
Taxi Data Analytics Results
Hotspot Detection Module
Wait Time Estimation Module
Trip Extraction Module
Extracted Hotspots
Density-Based Clustering
Pickup/Dropoff Locations
Pickup/Dropoff Event Extraction
Predicted Passenger Wait Time
Learning Model Selection
Key Features and Labels
Key Feature Extraction
Extracted Trips
Trip Distance and Duration Verification
Identified Trip Candidates
Taxi Trip Identification
Historical Taxi Data
Taxi State Transition Knowledge
Buffered Real Time Taxi Data
Offline Historical Data Collection
3G/ GPRS
Online data Collection
Taxi-Side Data Collection
Mobile Data Terminal (MDT)

Fig. 1. Block Diagram of the taxi analytics and visualization System.

visualizing the critical analytics results, where both the historical and real time taxi data are utilized. The system mainly consists of three analytics modules and a 3D visualization module, and the analytics modules are used for wait time estimation for the passengers, taxi hotspot detection and taxi trip extraction respectively.

The system block diagram is shown in Fig. 1, and the general descriptions of the four modules are given as follows.

– *Wait Time Estimation Module*: the main objective of the module is to make the prediction on passenger wait time at different regions. We derive the novel and effective key features, which include the FREE taxi taken (FTT) probability and taxi booking ratio (TBR), from taxi data to build the predictive model. We divide the passenger wait time into four levels as the outputs of the predictive model, namely *severe* (above 10 mins), *long* (5 to 10 mins), *reasonable* (2 to 5 mins), and *short* (below 2 mins), which have been summarized in Table 1. We will elaborate the feature extraction, model training, and other key issues for this module in Section 4.

Table 1
Four Levels of Passenger Wait Time

Model Output	Short	Reasonable	Long	Severe
Passenger Wait Time (min)	Below 2	(2, 5]	(5, 10]	Above 10

– *Trip Extraction Module*: this module mainly identifies and extracts the taxi trip information from the large scale taxi data. One trip is typically a taxi's sub-trajectory with a sequence of specific taxi state transitions. We thus design a dedicated algorithm to conduct the trip extraction, which will be elaborated in Section 5. The extracted trips and their distributions may directly illustrate the citywide taxi operating patterns. Moreover, the extracted trip information, especially their origin and destination locations, also serves as the key inputs for the hotspot detection and other analytics tasks.

– *Hotspot Detection Module*: this module detects both the pickup hotspots and dropoff hotspots, and we mainly take the pickup hotspots as an example in this paper. Pickup hotspots are the frequent taxi pickup locations, which usually emerge

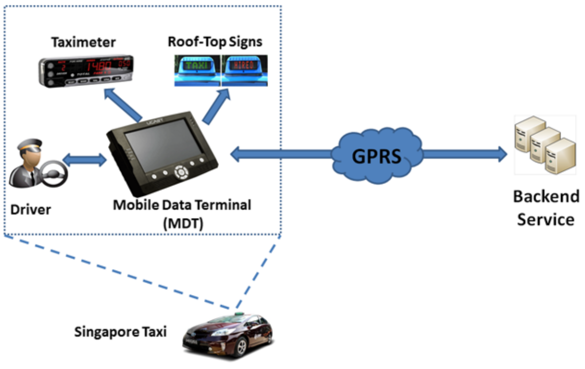

Fig. 2. A simplified telematics system with MDT on a Taxi.

when a high taxi demand occurs. The pickup hotspots may exist at any legal parking places and vary temporally and spatially. By leveraging on the extracted pickup locations, we design a specific algorithm to detect the taxi pickup hotspots, which will be described in Section 6.

– *3D Visualization Module*: this module mainly provides users an intuitive and interactive way to access and understand the analytics results from the above three modules. We adopt the 3D visualization techniques to dynamically depict the spatial and temporal information from multiple perspectives. To facilitate the interaction between the system and its users, the 3D map can be freely manipulated using mouse or touch screen with different control components. The details of this module will be given in Section 6 as well.

3.2. Data collection

As shown in Fig. 1, the system inputs are mainly the collected data from individual taxis. The taxi data collection is mainly leveraging on a specific device, called mobile data terminal (MDT), which has been installed on nearly all 26,000 taxis in Singapore. Figure 2 simply depicts a data collection system on a taxi, where MDT is hardwired directly to different on-vehicle devices, including taxi meter, roof-top signs. Moreover, it also provides taxi drivers a multifunctional touch screen to manually input and receive other key information, such as the taxi booking information.

During a taxi's daily operation, the MDT keeps collecting taxi's real time GPS locations, speed, and taxi states. The typical taxi states include FREE (available for passenger), ONCALL (booked by passenger), POB (passenger on board), and PAYMENT (passenger making payment). All the data collected by MDT can be

Table 2

Selected Fields of MDT Message with a Sample

Timestamp	Taxi ID	Longitude	Latitude	Speed	Taxi State
01/08/2016 19:04:51	SH0001A	103.7999	1.33795	54	POB

transmitted to the backend database either online or offline, where the online transfer is mainly relying on the 3G or GPRS communication. Table 2 gives the selected fields in an MDT message, which consists of 6 fields: timestamp, taxi ID, GPS location, instantaneous taxi speed, and taxi state. More details of the MDT system and the related information can be found in [24].

4. Wait time estimation module

It is a challenging problem to estimate the passenger wait time, as the passenger behaviors are not directly observable from taxi data at any given region. Therefore, it is necessary to find out some effective features that can help to infer passenger wait time. By observing the taxi operating and passenger waiting behaviors, multiple features are proposed and derived, where two key features are most effective, namely free taxi taken (FTT) probability and taxi booking ratio (TBR).

Briefly speaking, FTT probability captures how fast an available taxi is taken in a given region, where a large FTT value indicates that it is relatively easy for a FREE taxi to meet passengers in that region. TBR captures the fact that passengers would make a booking for taxi when it is hard to hail down a FREE one on the street. Thus, a high booking ratio is very likely caused by the long wait time of passengers in that region. In this section, we firstly derive the above two features, and then introduce the built model for passenger wait time estimation.

4.1. FREE taxi taken (FTT) probability

FTT probability is mainly motivated by the idea that a free taxi would have a high probability to meet a passenger when the high taxi demand coexists with the low taxi supply at a given region. Figure 3 shows a FREE taxi enters and exits a region with different supply and demand situations, where the green color and red color denotes FREE and POB, respectively.

Assuming a number of waiting passengers are uniformly distributed inside a given region, while a FREE taxi enters the region and keeps cruising. The larger number of waiting passengers means the higher den-

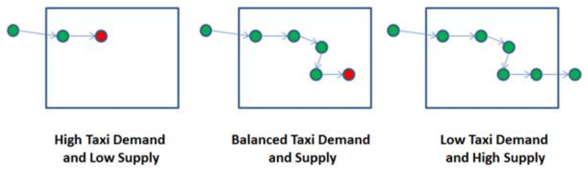
High Taxi Demand and Low Supply
Balanced Taxi Demand and Supply
Low Taxi Demand and High Supply

Fig. 3. A FREE Taxi under Different Situations.

sity of the taxi demand, and consequently the FREE taxi will meet its passengers in a shorter time period. The FTT probability is used to capture how fast a random taxi can be taken by a passenger in the region: let p be the probability that a FREE taxi meets a passenger within one time unit (e.g., one minute); meanwhile, we define a 2-element tuple, denoted as $\langle x, y\rangle$, for the same taxi, where x is the number of time unit that taxi keeps FREE, and y is a binary variable either 1 or 0, where $y = 1$ if taxi status turns into POB inside the region, and $y = 0$ if the taxi turns into other state (except POB) or leaves the region with the original FREE state.

Briefly speaking, the variable x and y are used to jointly describe a taxi behavior in a given region: x describes how long a taxi available inside the region, and y describes whether a taxi gets a passenger inside that region. For example, if a taxi keeps FREE for 5 minutes and gets a passenger at the 6th minute, $\langle x, y\rangle$ would be set to $\langle 5, 1\rangle$; if a taxi keeps FREE for 5 minutes and leave the region at the 6th minute without any passenger, $\langle x, y\rangle$ would be set to $\langle 5, 0\rangle$. Each taxi inside the region would have such a pair of x and y, and we use this information to estimate the FTT probability for that region.

Accordingly, we have the probability of a FREE taxi with $\langle x, y\rangle$ conditioning on p:

$$\Pr\big(\langle x, y\rangle | p\big) = (1-p)^x p^y.$$

Given D as all the tuples derived from the buffered real time taxi data, i.e., $D = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_n, y_n\rangle\}$, we have the likelihood function of p:

$$L(p|D) = \prod_{i=1}^{n} (1-p)^{x_i} p^{y_i},$$

and the corresponding log-likelihood function

$$\log L(p|D) = \left(\sum_{i=1}^{n} x_i\right) \log(1-p) + \left(\sum_{i=1}^{n} y_i\right) \log p.$$

We then maximize its log-likelihood function by letting its derivative to be zero:

$$\frac{d(\log L(p|D))}{dp} = \frac{-\sum_{i=1}^{n} x_i}{1-p} + \frac{\sum_{i=1}^{n} y_i}{p} = 0,$$

and hence we have the estimation of p:

$$p = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}.$$

Based on the above formula, we can calculate the FTT probability p for any region during a given time window. To help understand the above formula, we firstly show two extreme cases:

1. Suppose no passenger waiting inside the region during the given time window, all the FREE taxis would NOT change to the POB state. Hence, $y_i = 0$ for any i, which leads to $p = 0$;
2. Suppose a large number of passengers waiting inside the region during the given time window, any FREE taxi would quickly change to the POB state whenever it enters the region. Hence, $x_i = 0$ for any i, which leads to $p = 1$.

Calculation of the FTT probability p requires both x and y, and thus the system needs to extract the following information from the taxi data during the given time window: Taxi ID, T1, T2 and final taxi status. Taxi ID is the unique identifier of the taxi in the region; T1 is the first timestamp when the taxi is observed with the FREE state inside the region; T2 is the first timestamp when the taxi is observed with the non-FREE state (e.g., POB, ONCALL, etc.) inside the region, or the first timestamp when the taxi is observed with the FREE state outside the region (meaning the taxi leaves the region already without getting a passenger).

Figure 4 shows the simplified flowchart of the above described information extraction: when a new taxi record is received, the system checks whether the taxi is the first time appearing in the region. If so, the system checks whether the taxi is FREE or not. If FREE, the corresponding timestamp would be recorded as T1. If the taxi changes to FREE from another state inside the region, the corresponding timestamp would be recorded as T1 as well. After that, the system keeps monitoring the taxi location and state until it becomes non-FREE or leaves the region. Whenever such events occur, the corresponding timestamp would be record as T2. Finally, the system output would be $\langle ID, T1, T2, POB\rangle$ (taxi gets a passenger at T2), or

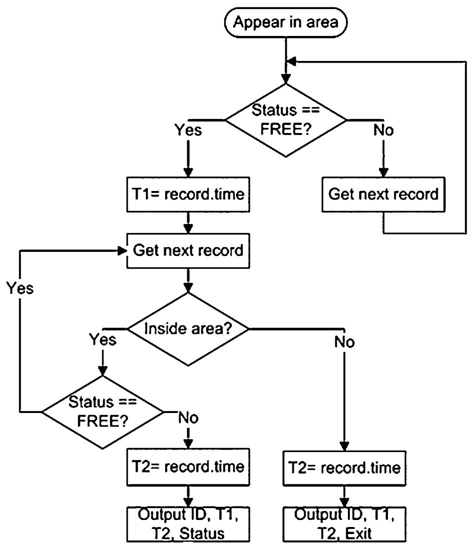

Fig. 4. Simplified Flowchart of the Information Extraction.

$\langle ID, T1, T2, EXIT\rangle$ (taxi leaves the region with FREE at T2), or $\langle ID, T1, T2, OTHER\rangle$ (taxi turns into other state at T2). Note that one taxi may generate multiple output records, as it may enter the same region multiple times. The built system is able to concurrently monitor multiple regions and process data from more than 20,000 taxis, where the regional geographical information is indexed in an in memory R-tree to speed up locating taxis (e.g., checking whether a taxi is currently inside the region or not).

In short, the FTT probability is not simply based on the discrete taxi pickup events, but a sequence of taxi cruising behavior and taxi free time length in a region. By aggregating such information from a large number of taxis, we can statistically make a good sense of the region's taxi demand-supply situation. The strong positive correlation between the FTT probability and the average passenger wait time has been validated, and more details can be found in our previous study [32].

4.2. Taxi booking ratio (TBR)

We observe that when a significant number of taxis are booked to pickup passengers in the same region, it is very likely that the taxi demand is much higher than supply there. Taxi booking usually means a passenger uses the local taxi operator's booking system to call a taxi. When a taxi driver accepts that booking job, the taxi state would change to ONCALL until the taxi successfully gets the passenger at the pickup location. After that, the taxi state would change to POB. Thus we can use $ONCALL \rightarrow POB$ transition and its location to count the taxi booking number at any given region. Moreover, our local driver behavior study shows that most of $BUSY \rightarrow POB$ transitions are caused by the taxi booking as well, and thus we jointly utilize both transition patterns to calculate the booking job numbers.

The absolute booking job number is hard to directly serve as an effective feature for the learning algorithms and models. We therefore further define the taxi booking ratio (TBR) instead of booking number as follows:

$$\frac{N(ONCALL \rightarrow POB) + N(BUSY \rightarrow POB)}{Total\ Number\ of\ Taxi\ Pickups},$$

where $N(\cdot)$ is the count number of the given transition.

Note that the BUSY status means the taxi driver temporarily unavailable. In practice, we found that this status is always used by taxi drivers in the same way as the ONCALL status, and thus we take the BUSY status with the ONCALL status together to count taxi booking number.

In short, TBR reflects the ratio of pickup number caused by taxi booking to the total pickup number at the given region and time period.

4.3. Recurrent neural network model

Based on the derived key features FTT probability and TBR, we compare multiple models and finally select the recurrent neural network (RNN) [34] to build the predictive model for passenger wait time. The feedback between RNN's hidden layers and input layer allows its hidden neurons to remember the history of the previously processed information. Therefore, it is suitable to process and build the model for the successive FTT and TBR features extracted from the temporally ordered taxi data. In recent years, the RNN architecture together with the long short-term memory (LSTM) [15] units has been widely used in a variety of tasks to support sequential data, and the GPU based computing architecture [31] significantly reduces RNN's training time.

In order to better identify the short term dependencies, each input of RNN at time slot T, say x_T, considers three consecutive and latest FTT probability values and TBR values, denoted as $x_T = \{F_{T-2}, F_{T-1}, F_T, R_{T-2}, R_{T-1}, R_T\}$, where F and R

Table 3
4-Element Vector of RNN Output

CLASS	Passenger Wait Time Range	4-Element Output Vector
1	Shorter than 2 Minutes	$\langle 0, 0, 0, 1\rangle$
2	Between 2 and 5 Minutes	$\langle 0, 0, 1, 0\rangle$
3	Between 5 and 10 Minutes	$\langle 0, 1, 0, 0\rangle$
4	Longer than 10 Minutes	$\langle 1, 0, 0, 0\rangle$

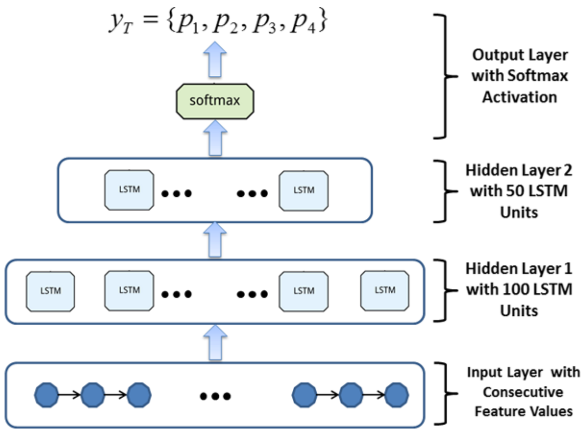
$y_T = \{p_1, p_2, p_3, p_4\}$
softmax
Output Layer with Softmax Activation
LSTM
Hidden Layer 2 with 50 LSTM Units
Hidden Layer 1 with 100 LSTM Units
Input Layer with Consecutive Feature Values

Fig. 5. Simplified Architecture of the Built RNN Model.

represent FTT and TBR respectively. Given y_T is the average passenger wait time at time slot T, we thus have an individual training data $\{x_T, y_T\}$, and the time window can be shifted by one time slot at each RNN time step. As mentioned earlier, we divide the passenger wait time into four levels, and thus y_T has four possible values, using a 4-element vector to represent, as given in Table 3. Note that the given four-level classification and its corresponding wait time ranges are mainly based on a local survey on taxi passengers in Singapore, while other ways of classification on taxi wait time can also be considered for other cities or for different time periods in a day (e.g., peak hours and off-peak hours).

The built RNN model consists of two hidden layers with 50 and 100 neurons respectively. Both of the two hidden layers adopt LSTM units. For the output layer, we use the softmax function as its activation. As the 4-element output vector, each element in the vector falls in the range of (0, 1), and the maximum one will be the predicted class. For example, the output $\langle 0.01, 0.07, 0.28, 0.64\rangle$ means the the prediction is class 1, i.e., passenger wait time shorter than 2 mins, as 0.64 is the largest element. The multiclass version of the log-likelihood loss is used as the loss function and RMSprop [33] is used as the optimizer. Figure 5 shows the simplified architecture of the built RNN model. Note that both features, namely FTT probability and TBR, are the region-based variables, the built RNN model is a region-specific model that can predict passenger wait time on any given region.

The passenger wait time data is collected by the relevant government agency. In Singapore, land transport authority (LTA) strives to collect and supply such information to the public commuters and taxi drivers. Currently, at more than 40 taxi stands and main pickup locations, LTA is conducting the daily survey by manually recording the wait time of passengers, and publishes the latest results on its official website.[1]

5. Trip extraction module

Taxi trip is typically a sub-trajectory that starts with a pickup event and ends with a dropoff event. This module mainly extracts the trip information from the raw taxi data, and the extracted trips would directly help to study and visualize the citywide taxi operating patterns. Moreover, the trip start and end locations can be used to further investigate taxi pickup, dropoff hotspots, and other operation related issues.

5.1. Preliminary and terms

We firstly define the important terms and expressions to be used in the following parts.

Definition 1 (Individual taxi's trajectory z)**.** A temporally ordered sequence of the MDT records from one taxi, i.e., $p_1 \rightarrow \cdots \rightarrow p_i \rightarrow \cdots \rightarrow p_n$, where p_i $(1 \leqslant i \leqslant n)$ is the ith record containing the taxi state $p_{i.\text{state}}$, instantaneous speed $p_{i.\text{speed}}$, latitude coordinate $p_{i.\text{lat}}$, longitude coordinate $p_{i.\text{lon}}$ and timestamp $p_{i.\text{ts}}$.

Definition 2 (Multiple taxis' trajectory set Z)**.** A collection of the individual taxi's trajectories, i.e., $\text{Z} = \{z^j \mid j = 1, 2, \ldots\}$, where z^j is the jth taxi's individual trajectory.

[1]http://www.lta.gov.sg/content/ltaweb/en/public-transport/taxis/taxis-and-the-lta.html

Algorithm 1 Taxi Trip Extraction Algorithm

```
Require: Taxi trajectory set Z, the thresholds $\eta_{\text{duration}}$, $\eta_{\text{sps}}$ and $\eta_{\text{spu}}$.
Ensure: Taxi trip set $P$.
 1: $k \leftarrow 1$; $R^k \leftarrow \emptyset$;
 2: for each individual taxi's trajectory $z$ in Z do
 3:   for $i = 2 \rightarrow n$ do
 4:     if $p_{i-1.\text{state}} = \textit{Non-POB}$ and $p_{i.\text{state}} = \textit{POB}$ and $R^k = \emptyset$ then
 5:       $R^k.Add(p_i)$; $R^k.Add(p_{i+1})$;
 6:     else if $p_{i.\text{state}} = \textit{POB}$ and $R^k \neq \emptyset$ then
 7:       $R^k.Add(p_i)$;
 8:     else if $p_{i.\text{state}} = \textit{Non-POB}$ and $R^k \neq \emptyset$ then
 9:       $R^k.Add(p_i)$; $k \leftarrow k+1$; $R^k \leftarrow \emptyset$;
10: for each $R^k \neq \emptyset$ do
11:   if time duration of $R^k < \eta_{\text{duration}}$ then
12:     Remove $R^k$;
13:   else if average speed of $R^k < \eta_{\text{sps}}$ then
14:     Remove $R^k$;
15:   else if average speed of $R^k > \eta_{\text{spu}}$ then
16:     Remove $R^k$;
17:   else
18:     $P.Add(R^k)$;
19: Output the extracted sub-trajectory set $P$;
```

5.2. *Taxi trip extraction*

In order to extract taxi trip from the taxis' trajectory set Z, we propose a simple and practical algorithm, called taxi trip extraction (TTE) algorithm, which is shown in Algorithm 1.

The TTE algorithm in general consists of two steps: firstly, given the taxi trajectory set Z, it identifies the taxi sub-trajectories with the certain state transitions and keeps them as the trip candidates: namely from Non-POB state to POB state and return to Non-POB again. Secondly, it filters out the trip candidates that have too short duration or abnormal average speed, as such sub-trajectories are normally caused by the MDT internal error or taxi driver's improper operations on taximeter or MDT. The algorithm's final output P is the extracted trip set, which consists of multiple taxi sub-trajectories. Each sub-trajectory represents one taxi trip, including the information from the pickup event until the dropoff event.

6. Hotspot detection and visualization

6.1. *Hotspot detection module*

As mentioned earlier, hotspot can be either pickup hotspot or dropoff hotspot. In this paper, we mainly

Algorithm 2 Pickup Hotspot Detection Algorithm

```
Require: Taxi trip set $P$ and region $\Omega$.
Ensure: Pickup hotspot set $Q_{\text{loc}}$.
1: for each $R^k$ in trip set $P$ do
2:   Extract the first POB location from $R^k$, say $c^k$;
3:   if $c^k$ outside the given region $\Omega$ then
4:     Remove $R^k$;
5:   else
6:     Add $c^k$ into the location set $H$;
7: Run DBSCAN clustering algorithm on set $H$;
8: Add the centroid of the found clusters into $Q_{\text{loc}}$;
```

present and study the taxi pickup hotspot, but similar methodology and algorithm can be applied to the dropoff hotspot. Pickup hotspots are the frequent taxi pickup locations, which are normally located at the places with a high taxi demand, such as the main entrance of hotels, hospitals, and schools. The pickup hotspots may dynamically change and vary temporally and spatially. Study and visualization of pickup hotspots would directly help to understand the taxi demand distribution and driver pickup behaviors. We thus propose a specific algorithm, called pickup hotspot detection (PHD) algorithm, to identify the hotspots. It is mainly based on the extracted taxi trips, and the main steps are given in Algorithm 2.

The PHD algorithm utilizes the extracted trip set P as its main input, and outputs the pickup hotspot set Q_{loc}. Given all the trips in P and region Ω, the algorithm firstly filters out the trips start outside Ω, and then for the leftover trips, adds their first *POB* locations into the location set H. After that, the algorithm runs the density based clustering algorithm called *DBSCAN* on set H, and computes the centroid of each found clusters. All the computed centroids would be added into the final output Q_{loc}, namely the identified taxi pickup hotspots. Note that hotspot detection normally requires a relatively long period of taxi data, and thus the module may use both the historical and buffered taxi data.

The PHD algorithm adopts the clustering algorithm DBSCAN [13], which is an effective way to discover high density clusters and meanwhile remove noises. When running DBSCAN with the PHD algorithm, its parameters need to be carefully selected. We will address the parameter selection issue in the following evaluation section. On the other hand, other advanced density based clustering methods [35] can be considered and applied in the PHD algorithm as well. Moreover, the proposed hotspot detection is region-specific, and different region Ω and different parameters for clustering can be set in Algorithm 2.

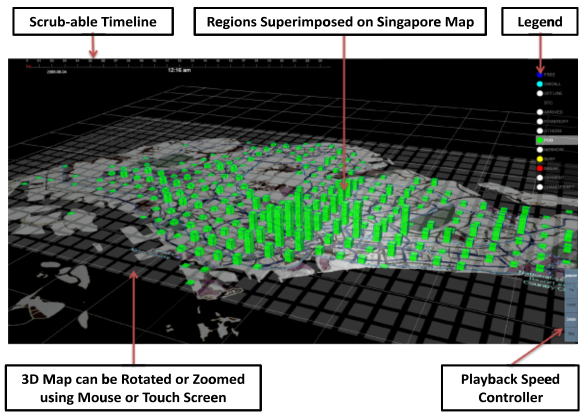
Scrub-able Timeline
Regions Superimposed on Singapore Map
Legend
3D Map can be Rotated or Zoomed using Mouse or Touch Screen
Playback Speed Controller

Fig. 6. User Interface of the Visualization Module.

6.2. 3D visualization module

To provide system users an intuitive and interactive way to access and understand the analytics results, we implement the 3D visualization module and its user interface [22]. Given the spatial and temporal characteristics of the analytics results, the visualization module depicts the spatial information by superimposing it on Singapore map, and the spatial information would dynamically change with the temporal information. Figure 6 shows a snapshot of the built 3D visualization. The module also provides the scrubbable time line on the upper side and the playback speed controller on the bottom right corner. The system users can select the starting time by simply clicking or dragging the scrubbable time line, and the playback speed controller allows for setting different speed as well as pausing the streaming data, which is similar as a video player control panel. The 3D map can be freely rotated and zoomed in/out using either mouse or touch screen.

The visualization module is mainly implemented by HTML5, and can support different browsers, including Internet Explorer, Chrome and Firefox. The WebGL [18] is used for rendering the interactive 3D graphics and user interface.

7. System evaluation and visualization

7.1. Passenger wait time estimation

We trained the designed RNN model using a highly modular neural networks library Keras [7], which was written in Python and capable of running on top of Theano [5]. After fine-tuning the parameters, 10% dropout was used for the LSTM units and the batch size was set to 3. The RMSprop [33] was used as the optimizer, where its learning rate and rho were set to 0.001 and 0.9 respectively. The training process was conducted on the dedicated server with two NVIDIA GPUs (Titan X 12 GB GDDR5). The region we choose to build the model is the central business district (CBD) of Singapore, which has the heaviest and highly dynamic traffic, and thus it is usually the hardest task to estimate passenger wait time for that region.

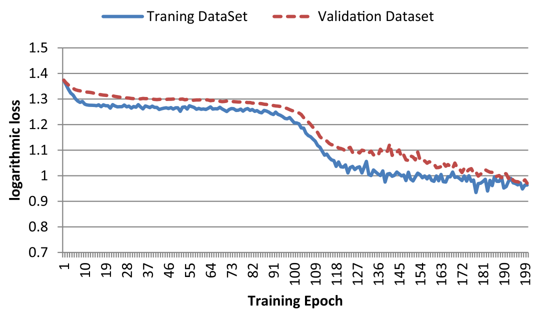

Fig. 7. Loss Function with Training Epoch.

Figure 7 showed the log-likelihood loss significantly drops down on both datasets after multiple training epoches. The overall classification accuracy was around 73.3%. We compared it with other two popular classification models: gradient boosting machine (GBM) and random forest (RF). Table 4 summarized the best accuracies the three models achieved on the validation dataset: it showed that RNN achieved the highest accuracy, which was probably because RNN's hidden layers successfully captured and linked the sequential features at the time domain. When building different models for estimation of passenger wait time, multiple features were introduced and used, while the most effective features were the FTT probability and TBR.

To verify the statistical significance of the above results, we adopt the two-tailed T-test [25] on the multiple runs of the ten-fold cross validation. The p-value of the mean accuracy of RNN model and RF model are 0.0000204, and the p-value of the mean accuracy of RNN mode and GBM model are 0.0000153. Table 5 and Table 6 further give other information of the significance test, including F statistic and the degree of freedom. The above results verify the model accuracies are significantly different, and thus the proposed RNN model performs better than another two classic models statistically.

Table 4
Overall Classification Accuracy with Different Models

Training Model	Overall Classification Accuracy
RNN	73.3%
Random Forest	67.5%
GBM	65.4%

Table 5
Statistics of Model Accuracy Comparison (RNN vs RF)

	RNN	RF
Variance	0.000691	0.000487
Degree of Freedom	9	9
F statistics	1.418192	
F critical (alpha = 0.05)	3.178893	

Table 6
Statistics of Model Accuracy Comparison (RNN vs GBM)

	RNN	GBM
Variance	0.000691	0.000411
Degree of Freedom	9	9
F statistics	1.680415991	
F critical (alpha = 0.05)	3.178893	

7.2. *Trip extraction and pickup hotspot detection*

We ran the TTE algorithm to extract the taxi trips, where the thresholds η_{duration}, η_{sps} and η_{spu} were set to 60 seconds, 1 km/hour, and 150 km/hour. The daily extracted trip number varied in terms of working day and weekend day, but the normal range was from 450,000 to 550,000. We used an independent taxi trip information kept in taximeter to validate the TTE extraction results. The trip start and trip end time together with the taxi ID were used to compare the trips, and it showed that around 97% trips were successfully identified and accurately matched.

We ran the PHD algorithm on the extracted trip data, and successfully detected the taxi pickup hotspots during the given time slots. The detected pickup hotspot number varied at different time slots. For example, Fig. 8 showed the detection result during an evening peak hour, and nearly 200 hotspots were found across the island, while Fig. 9 showed the detection result during an early morning hour, and only around 20 hotspots were found. To compare with the exiting studies, we also perform the hotspot detection method used in [36], which is mainly based on the OPTICS algorithm [2]. Only around 130 hotspots were successfully detected across the island using this method, which occupies around 65% of the hotspots detected by the proposed method.

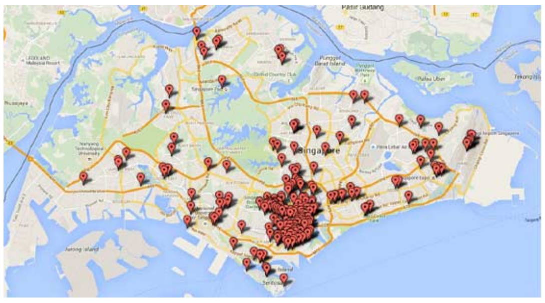
Fig. 8. Detected Pickup Hotspots during an Evening Peak.

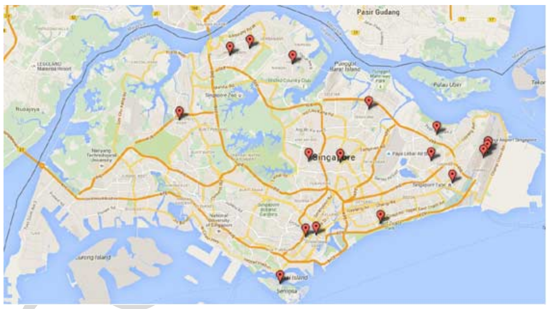
Fig. 9. Detected Pickup Hotspots during an Early Morning.

For the PHD algorithm, running the DBSCAN clustering algorithm on a large size location set was computationally expensive due to its $O(n^2)$ complexity. Moreover, properly choosing the two parameters of DBSCAN, i.e., eps ε_d and min-points p_d, was not a trivial issue: ε_d specified the maximum radius of the neighborhood and p_d sets the minimum number of points in an eps-neighborhood of the point. An unduly small ε_d or an overly large p_d might lead a large number of the data points cannot be clustered, while an overly large ε_d or an unduly small p_d would merge different clusters into one. Figure 10 showed the number of the detected hotspot with respect to different ε_d and p_d. We saw that small ε_d or large p_d values result that only a few number of hotspots were detected and many actual ones were neglected. On the other hand, large ε_d values or small p_d values would easily merge adjacent hotspots and meanwhile bring the insignificant hotspots. By carefully study such parameters, we usually set ε_d to 15 meters. For the parameter p_d, different values were used for different time durations: for example, we set p_d to 10 points when processing 15 mins buffered time period, while we set p_d to 50 points when processing the daily taxi data. In short, the DBSCAN parameters needed to be carefully selected and tuned.

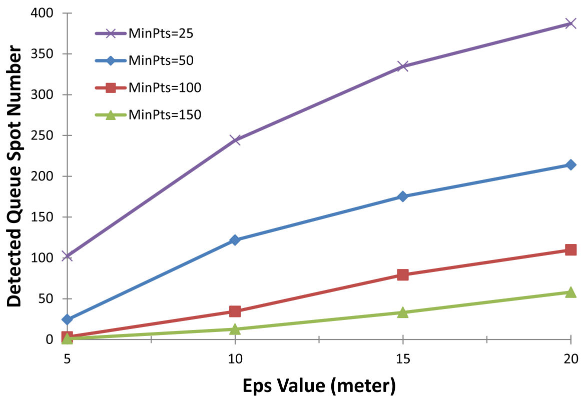

Fig. 10. DBSCAN Performance with Different Parameters.

In short, the three analytics modules were mainly implemented in JAVA and Python, and can be invoked manually from the user interface or automatically based on the system configuration file. The JETTY server [12], which supplied a java servlet container and a HTTP server, was selected to host them. The taxi data and the analytics results were stored in two separated databases.

7.3. Visualization

We showcased some snapshots of the built visualization module, especially the outputs from the three analytics modules, namely wait time estimation, trip extraction and hotspot detection.

Figure 11 showed the estimated passenger wait time at the lunch time (around 12:30PM) in a working day. We saw that the four colors, representing the four wait time ranges, depict the different predefined regions. The three regions on the Singapore west (the left side of the map) were all in red (more than 10 mins), which was probably due to the low taxi supply in those remote areas. Many regions were in green (less than 2 mins) or light green (between 2 and 5 mins) colors, showing a better balance between taxi supply and demand there.

Figure 12 showed the estimated passenger wait time in the evening (around 10PM) in the same working day. We saw that the downtown area (bottom of the map) were in either red or orange (between 5 to 10 mins) colors, which was probably due to the high taxi demand in that area. Meanwhile, most of other regions were in green probably due to low taxi demand at night.

On the built 3D map, each region was selectable and can be highlighted to dynamically show the variance of passenger wait time. Moreover, the visualization module allowed the system users adjust each region and change its update frequency.

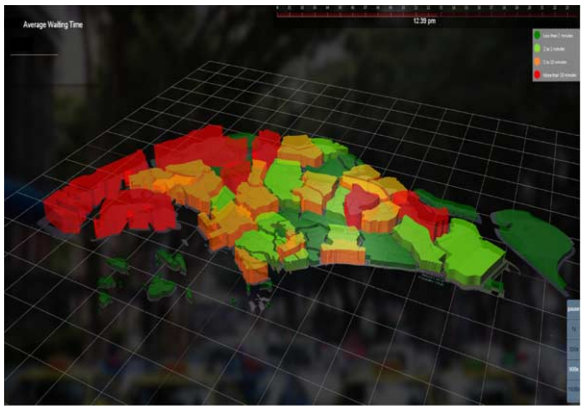

Fig. 11. Predicted Passenger Wait Time at Noon.

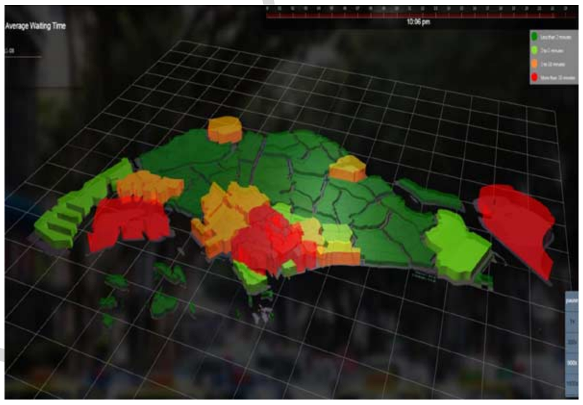

Fig. 12. Predicted Passenger Wait Time at Night.

Figure 13 showed the extracted trips at the Singapore Changi airport region between 2:00AM and 2:15AM, where the green lines represented the outgoing trips and the pink lines represent the incoming trips. We saw that most lines were in green and a few in pink, showing only a few number of taxis taking passengers to the airport during that time period.

Figure 14 showed the extracted trips at the downtown region between 7:15PM and 7:30PM, where the total trip number was apparently larger than Fig. 13 and most lines were still in green. It was probably caused by people back home from their office during that period.

On the built 3D map, the users can freely select any region to observe and highlight its incoming/outgoing trips based on the trip extraction results.

Figure 15 showed the detected pickup hotspots during an evening peak (from 7PM to 7:30PM). We saw that the downtown area (bottom of the map) gathers

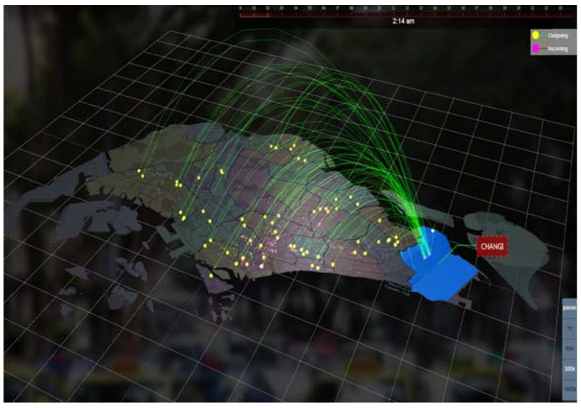

Fig. 13. Incoming and Outgoing Trips at Changi Airport Region.

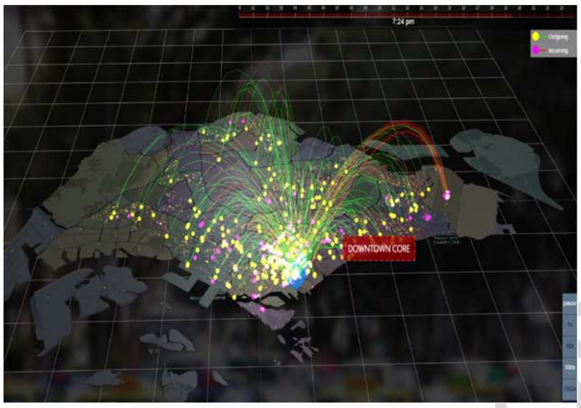

Fig. 14. Incoming and Outgoing Trips at Downtown Region.

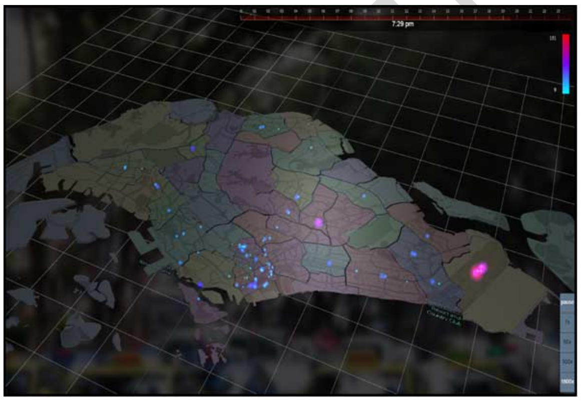

Fig. 15. Detected Hotspots with the Cluster Density.

a significant number of the detected hotspots. Meanwhile, several hotspots, located at the center of the map and Changi airport, were in red color, which means the pickup frequencies there were pretty high.

8. Conclusion

In this paper, we presented a system for taxi service analytics and visualization. It contained the three analytics modules and a 3D visualization module. The RNN based model was proposed and implemented to estimate the regional passenger wait time, which utilizes several derived key features, including FTT probability and taxi booking ratio. The built model showed a better accuracy than other popular learning models. Moreover, we designed the novel and practical algorithms to successfully detect hotspots and extract taxi trips. We further designed the frontend 3D visualization and interactive user interface for the three analytics modules. The 3D visualizations provided system users an effective way to access and understand the taxi operation patterns from the analytics results. The experimental results show that the built RNN-based predictive model achieves 73.3% overall accuracy, which is significantly higher than other classic models. Meanwhile, around 97% trips are accurately identified and more than 200 hotspots in the city are successfully detected.

To the best of our knowledge, it is the first work applying the RNN-based model to conduct the taxi passenger wait time estimation. Meanwhile, a novel 3D visualization solution with an informative user interface is designed to effectively help users to access the key taxi analytics results, including trips and hotspots. While the current system archives our design objectives, how to predict the passenger wait time in a better accuracy is still an open and interesting research problem. For example, in addition to the data from the taxi side, new information can be introduced to derive new features for the prediction model, such as the weather information and the crowd-sourced passenger mobility information. Moreover, we are planning to periodically publish the estimated passenger wait time and hotspots online to help the public and other stakeholders in Singapore, including taxi companies and government agencies.

Acknowledgement

This research is partially supported by the National Natural Science Foundation of China (No. 61702039).

References

[1] G. Andrienko, N. Andrienko and S. Wrobel, Visual analytics tools for analysis of movement data, *SIGKDD Explor. Newsl.* **9**(2) (2007), 38–46. doi:10.1145/1345448.1345455.

[2] M. Ankerst, M.M. Breunig, H.-P. Kriegel and J. Sander, Optics: Ordering points to identify the clustering structure, in: *ACM Sigmod Record*, Vol. 28, ACM, 1999, pp. 49–60.

[3] J. Aslam, S. Lim, X. Pan and D. Rus, City-scale traffic estimation from a roving sensor network, in: *Proc. ACM Conference on Embedded Netw. Sensor Sys. (SenSys)*, 2012.

[4] Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks* **5**(2) (1994), 157–166. doi:10.1109/72.279181.

[5] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio, Theano: A cpu and gpu math compiler in python, in: *Proc. 9th Python in Science Conf.*, 2010, pp. 1–7.

[6] P.S. Castro, D. Zhang, C. Chen, S. Li and G. Pan, From taxi gps traces to social and community dynamics: A survey, *ACM Comput. Surv.* **46**(2) (2013), 17:1–17:34. doi:10.1145/2543581.2543584.

[7] F. Chollet, Keras: Deep Learning library for Theano and TensorFlow [online]. Available: https://github.com/fchollet/keras.

[8] A. De Brébisson, E. Simon, A. Auvolat, P. Vincent and Y. Bengio, Artificial neural networks applied to taxi destination prediction, in: *International Conference on ECML PKDD Discovery Challenge, ECMLPKDDDC'15*, 2015.

[9] P. Doetsch, M. Kozielski and H. Ney, Fast and robust training of recurrent neural networks for offline handwriting recognition, in: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, 2014, pp. 279–284.

[10] J. Dykes, A. MacEachren and M. Kraak, Impact of data and task characteristics on design of spatio-temporal data visualization tools, *Exploring Geovisualization* (2005), 201–222.

[11] J. Dykes, A. MacEachren and M. Kraak, Visualizing, querying and summarizing individual spatio-temporal behaviour, 2005.

[12] Eclipse Foundation [online]. Available: http://www.eclipse.org/jetty/.

[13] M. Ester, H.P. Kriegel, J. Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

[14] S. Guha, R. Rastogi and K. Shim, Cure: An efficient clustering algorithm for large databases, in: *ACM Sigmod Record*, Vol. 27, ACM, 1998, pp. 73–84.

[15] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation* **9**(8) (1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[16] T. Kapler and W. Wright, Geotime information visualization, *Information Visualization* **4**(2) (2005), 136–146. doi:10.1057/palgrave.ivs.9500097.

[17] A. Keler and J.M. Krisp, Is there a relationship between complicated crossings and frequently visited locations? – a case study with boro taxis and osm in nyc, in: *Proc. International Conference on Location-Based Services*, 2016.

[18] Khronos Group, WebGL 2 Specification [online]. Available: https://www.khronos.org/registry/webgl/specs/latest/2.0/.

[19] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi and Q. Yang, Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset, in: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, IEEE, 2011, pp. 63–68. doi:10.1109/PERCOMW.2011.5766967.

[20] S. Liu, Y. Liu, L.M. Ni, J. Fan and M. Li, Towards mobility-based clustering, in: *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[21] S. Liu, N. Yang, M. Li and M. Zhou, A recursive recurrent neural network for statistical machine translation, in: *ACL (1)*, 2014, pp. 1491–1500.

[22] Y. Lu, G.G. Chua, H. Wu and C.S.Q. Ong, An intelligent system for taxi service monitoring, analytics and visualization, in: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016, pp. 4256–4257.

[23] Y. Lu, A. Misra, W. Sun and H. Wu, Smartphone sensing meets transport data: A collaborative framework for transportation service analytics, in: *IEEE Transactions on Mobile Computing*, 2017.

[24] Y. Lu, S. Xiang and W. Wu, Taxi queue, passenger queue or no queue? – a queue detection and analysis system using taxi state transition, in: *Proc. International Conference on Extending Database Technology (EDBT)*, Belgium, 2015.

[25] R. Mankiewicz, *The Story of Mathematics*, Cassell, 2000.

[26] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira and L. Damas, Predicting taxi-passenger demand using streaming data, *IEEE Transactions on Intelligent Transportation Systems* **14**(3) (2013), 1393–1402. doi:10.1109/TITS.2013.2262376.

[27] R.T. Ng and J. Han, Clarans: A method for clustering objects for spatial data mining, *IEEE transactions on knowledge and data engineering* **14**(5) (2002), 1003–1016. doi:10.1109/TKDE.2002.1033770.

[28] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman and C. Ratti, Taxi-aware map: Identifying and predicting vacant taxis in the city, in: *International Joint Conference on Ambient Intelligence*, Springer, 2010, pp. 86–95. doi:10.1007/978-3-642-16917-5_9.

[29] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko and G. Andrienko, Visually driven analysis of movement data by progressive clustering, *Information Visualization* **7**(3–4) (2008), 225–239. doi:10.1057/PALGRAVE.IVS.9500183.

[30] H. Sak, A. Senior and F. Beaufays, Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014, arXiv preprint arXiv:1402.1128.

[31] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, 2010.

[32] D. Shao, W. Wu, S. Xiang and Y. Lu, Estimating taxi demand-supply level using taxi trajectory data stream, in: *Proc. IEEE International Conference on Data Mining (ICDM) Workshop*, 2015.

[33] T. Tieleman and G. Hinton, Neural Networks for Machine Learning, Techical report, 2012.

[34] R.J. Williams, Training recurrent networks using the extended Kalman filter, in: *Neural Networks, 1992. IJCNN, International Joint Conference on*, Vol. 4, 1992, pp. 241–246.

[35] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, 2005.

[36] N.J. Yuan, Y. Zheng, L. Zhang and X. Xie, T-finder: A recommender system for finding passengers and vacant taxis, *IEEE Transactions on Knowledge and Data Engineering* **25**(10) (2013), 2390–2403. doi:10.1109/TKDE.2012.153.

[37] F. Zhang, D. Wilkie, Y. Zheng and X. Xie, Sensing the pulse of urban refueling behavior, in: *Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing(UbiComp)*, 2013.

[38] Y. Zheng, Trajectory data mining: An overview, *ACM Transactions on Intelligent Systems and Technology (TIST)* **6**(3) (2015), 29.