Bus Transit Time Prediction using GPS Data with Artificial Neural Networks

Fan Jiang

Wednesday 29th November, 2017

1 Extended Abstract

Background. Bus transit time prediction is to predict how long it takes for a bus to travel from point A to point B. Good prediction helps with urban transportation planning and bus riders' time planning. Artificial neural networks have been proven to work well in this field.
Aim. This project aims to make good travel time predictions for both route and segments of route with the bus GPS data using artificial neural networks.
Data. We use a collection of 925,368 latitude and longitude data points taken for buses in Porto, Portugal from April 5th to April 7th, 2015. We also used route map and bus stops data from Porto bus transportation website (`http://www.stcp.pt/en/travel/`) as prior knowledge. The bus GPS dataset and prior knowledge are used together to compute features for the models.
Methods. We developed 3 models to make the travel time prediction, two of which are to predict whole route travel time and the other one is to predict segment of route travel time. All three models use a three-layer neural network, with different number of input and hidden units. The first model predicts the whole route travel time with features computed directly from bus GPS data and prior knowledge, the second model predicts segment of route travel time with preceding bus information and the last model predicts the whole route travel time by combining predictions of segments.
Results. As the model takes more prior knowledge, higher accuracy is achieved. Moreover, whole route travel time prediction using segments has better results than the route prediction model using solely bus GPS data. It also performs better than the segment prediction model, because negative and positive errors of segment predictions cancel each other out.
Conclusions. First, it is promising to develop a model that takes as many factors as possible to make accurate travel time predictions, given that prior knowledge improves model accuracy. Second, when there is lots of variations in training set, neural network may not be a good option because it tries to "overfit" outliers.
Intellectual merit. This project can serve as a basis for the development of an advanced public transportation system, which is able to make highly accurate bus transit time predictions using bus GPS data, prior knowledge and real-time traffic.
Broader Impacts. With more accurate route travel time predictions, the transportation planning institute is able to arrange more efficient bus schedules and design better bus routes. As for individuals, better segment travel time predictions can provide much convenience and efficiency on time planning.

2 Problem Statement

The main idea of this project is to predict the bus travel time using bus GPS data with the help of other information such as bus routes and bus stops information. The models should make good predictions on route travel time as well as segments of route travel time using the three-layer artificial neural networks.

3 Introduction

Bus transit has been a basic yet vital way of transportation in most cities. From government's point of view, bus transit can be easily accessed by mass people. It is a timely reliable, cheap to ride, all-year-along operational and cheap-to-maintain transportation. From a city dweller's point of view, bus is more economical and eco-friendly than driving a car.

Because of the advent of bus-only lanes in many cities, bus is even faster than car in some traffic-heavy areas during rush hours. Although bus is not always on schedule as subway, bus usually covers more areas than subway does. Therefore, bus has been top-one choice of transportation in many cities for people like daily commuters or travelers. It comes as no surprise that bus transit planning is an important part of urban transportation planning, and that advanced public transportation systems (APTS) and advanced traveler information systems (ATIS) have been developed over the years to provide convenience for bus riders. Hall et al. (1997) pointed out that intelligent transportation systems provides modest benefits in reducing passenger delay - on the order of 20 seconds per transferring passenger on average. Related technologies like global positioning systems (GPS), automatic vehicle location systems (AVLS) and automatic passenger counters systems (APCS) have expanded quickly to support the bus transit systems. And two research problems have drawn most attentions: bus transit time prediction for a single route and bus arrival time prediction.

However, it is not easy to accurately predict bus transit time, because unlike subway, bus transit time is affected by many external factors, such as weather and road traffic. An ideal model would be able to take as many factors as possible to make precise predictions. In addition, historical data is not enough to make dynamic predictions. It is desirable to dynamically capture real-time transit-related information while making predictions. Third, in cities with developed bus transportation, it is normal for multiple routes to use common bus stops. Using information from different routes not only implicitly embeds traffic information, but also provides bus riders with more options of routes to destination. In this project, we will explore the effects of adding more factors to the model when predicting the bus travel time. Moreover, we will use multiple routes information, historical bus GPS data and bus routes prior knowledge to predict bus transit time. All models utilize a three-layer neural network. A three-layer neural network is a good choice for this project because of three reasons: First, by universal approximation theorem, a three-layer feed forward neural network containing a finite number of hidden neurons can approximate any continuous functions on compact subsets of R^n, under mild assumptions on the activation function; Second, a three-layer neural network is enough to solve a large number of problems in real life. Third, more hidden layers will likely overfit the training set. Especially in this project, the size of training data is small.

4 Related Work

Bus Arrival Time Prediction with GPS Data
Lin and Zeng (1999) uses bus GPS data obtained from automatic vehicle location systems, together with other information - bus schedule table, bus delay and time check point - in algorithms to predict bus traveling time from one bus stop to another. The two-dimensional latitude-longitude data was transformed into a one-dimensional bus path with link-node representation. Then the data is assessed on its accuracy, on-time performance and delay correlation. In the paper, base case algorithm uses bus schedule table as prediction. In the other four algorithms, the paper starts with bus GPS data, additively including bus schedule table, delay and time check point. All five algorithms are compared with overall precision, robustness and stability. It turns out that time check information has the best influence on bus travel time.
Model Selection
Gurmu and Fan (2014) compared three models to predict bus travel time dynamically using only GPS data. The three models are historical average, kalman filtering and ANN. Three models are tested on GPS data for bus line LT11 in Macae, Brazil from November 2008 to May 2009. By analyzing the data, the authors found out that the travel time between stops vary a lot over time of the day, however, the travel time distribution over different days of the week seem to be nearly the same. Three sections on the trajectory of different lengths are used to test the models, and models are compared based on Mean Absolute Percentage Error (average derivation) and Maximum Absolute Percentage Error (max derivation / robustness). All three models do well for short sections, but kalman filtering and ANN outperform historical average for both medium and long sections. However, kalman filtering fails at giving reasonable estimation during peak hours, especially evening peak. On the other hand, ANN can give stable predictions during peak and off-peak hours. In addition, ANN makes most robust predictions and has the lowest prediction error overall standard deviation. The authors also found out that ANN is less effective in predicting bus travel time for very short and/or long trips. That is, only when the observed travel times are in a range between 20 to 50 minutes, the ANN is able to provide real-time travel information with less than 10% MAPE.
Bus Arrival Time Prediction using Multiple Routes
Yu et al. (2011) proposed a method to predict bus arrival time using multiple bus routes. Bus running time of different routes were used to estimate current traffic conditions. Four models were implemented and compared: Support Vector Machine (SVM), ANN, k-NN and Linear Regression (LR). The results showed that this proposed method is more accurate than the models based on bus running time of a single route. Moreover, SVM has the best performance, k-NN and ANN

tied for second places and LR was the worst among the four models. We replicate and explained the model in details in section 6.3.

5 Data

The dataset is a collection of global positioning system (GPS) points for buses in Porto, Portugal. Each data point records the bus' latitude, longitude and other 30 features of a bus at a time instant. The information is emitted by on board unit (OBU) on a bus and sent to the system through one of the three networks: 802.11p, physical and cellular network.

5.1 Datasets

There are two bus GPS datasets, with same features, but different sizes. The smaller dataset contains data points from April 5th to 7th, 2015, in total of 925,368 data points. The other dataset covers whole April, from 1st to 30th, and contains 2,300,742 data points. We will use both datasets to do bus travel time prediction. In particular, we will use the small dataset for all models and use the large dataset on some models to see if more data would improve prediction accuracy.

5.2 Data Features

Each point contains 32 features: node id, system time, gps time, server time, latitude, longitude, altitude, speed, heading, hdop, acceleration x, acceleration y, acceleration z, hops, network, next hop id, rsu id, rssi, vehicle status, distance, connected time p, connected time 3p, traffic 3g in, traffic 3g out, traffic g in, traffic g out, traffic p in, traffic p out, traffic eth in, traffic eth out, traffic 3g l2 in, and traffic 3g l2 out. In this project, we only used 4 features: node id, system time, latitude and longitude. Description of all feature fields can be found in Appendix. We took three examples

Feature Name	Description
node_id	ID of the installed boards in the vehicles
system_time	Time instance of the board, generally the same as GPS time.
latitude	Latitude of vehicle
longitude	Longitude of vehicle

Table 1: Description of used features.

from the dataset with only the four fields shown. Examples of full feature data point can be found in Appendix.

Feature	node_id	system_time	latitude	longitude
Example 1	195	2015-04-05 00:00:00	41.148689	-8.610642
Example 2	215	2015-04-05 00:00:00	41.173859	-8.619491
Example 3	372	2015-04-05 00:00:00	41.183071	-8.621082

Table 2: Selected Samples from data.

Node id is an integer that identifies each **bus vehicle**. **System time** has precision to minutes in the small dataset and to seconds in the larger dataset. Both **latitude** and **longitude** are high-precision doubles. In some cases, the latitude and longitude will both be 0. We regard this as a recording error and throw such data points away. We will generate feature spaces from these four data fields, and feed to travel time prediction neural networks.

5.3 Properties of Raw Data

5.3.1 Frequency of Data Recording

In the small bus dataset, the time ranges from 2015-04-05 00:00:00 to 2015-04-07 17:22:00, with duration of 235,320 seconds. During one minute, the OBU transmits bus information 3 to 4 times. If we divide the duration by the frequency of bus GPS data transmissions, we are expecting around 15,000 data points for each bus vehicle (node id). However, on average, each vehicle records only 1940 times. Maximum number of information transmissions for a bus is 10,843 and

minimum is 34. Vehicles don't transmit data all the time. It is normal for a bus to stop sending information a few times a day and each interval ranging from one hour to several hours.

5.3.2 Bus Trajectory

In the small dataset, we find 477 unique vehicles. Latitude and longitude for the same vehicle during an non-interrupted recording session are grouped together. Ordering them by system time, we get a **trajectory**. In the small dataset, there are usually two to three trajectories for each bus vehicle.

5.4 Prior Knowledge

Route is an ordered sequence of bus stops. Each **node id** must operate on a route. Our model is to predict the route travel time. We collect two kinds of routes information as prior knowledge and use them to map trajectories to routes.

5.4.1 Route

The first prior knowledge is a bus transportation map in Porto, Portugal. The map shows each route in blue lines and annotate the route id along the line. Parts of a blue line may have many route id annotated, because different routes can share common segments. The map is not detailed enough to show sequence of bus stops, however, it annotates several bus stop "hubs" that are passed by multiple routes and indicates if the stop is terminal for some routes. The map partitions the bus transportation system in Porto into 18 **zones**: C1, C2, C3, C4, C5, C6, C8, C9, C10, C11, C16, S1, S2, S8, S9, N10, N11, N16. Visually speaking, different zones have different density of bus routes. Downtown Porto lies in zone C1 and it is the most traffic-heavy zone. Figure 1 shows a small part of the map where zone C1 lies in.

Figure 1: Porto bus route map, routes shown in blue lines, route id annotated and bus stop hubs pointed out

5.4.2 Bus Stops

The second prior knowledge set is the route information on Porto's official bus transportation website (`http://www.stcp.pt/en/travel/`). There are 65 bus routes, 4 of which are circular and the rest are non-circular. Circular route has

only one direction and non-circular route has two directions. So there are 126 ordered sequences of bus stops. Each bus stop contains the GPS, stop name, stop code in capitalization, detailed address, and the zone number.

5.5 Processing of Raw Data

5.5.1 Trajectory Visualization

We plot bus trajectories on a map, as shown in Figure 2. Some plotting exhibits strong route patterns. We found out that a bus usually waits at a bus gathering place before starting the service. Then the vehicle drives from the gathering place to the route start stop, and repeats the route over and over. In the figure, certain parts of the trajectory is thicker than other parts, because the bus traverse these parts more times. Usually the thicker part of trajectory is the route and the lighter part is the path from the gathering place to the route starting point. From the visualization, we also find out trajectories of circular and non-circular routes.

5.5.2 Trajectory Snapping

The trajectory is plotted with polylines corresponding to sequences of GPS. Because of the high-precision nature and drifting effect of GPS, the trajectory cannot lie perfectly with the actual roads. Wang et al. (2014) proposed an algorithm that can reshape the odometry trajectory to fit the constraints of a given topological map. After applying the algorithm, we obtained clearer bus trajectories, as the contrast shown in Figure 3. We only use snapping for visualization purpose.

5.5.3 Node Id to Route Mapping

With the visualization of snapped trajectories and route prior knowledge, we can match trajectory to route. Starting with a trajectory, we can match it with segments on the map and identify the possible routes. We then check the routes on transportation website and match the route shapes. We only record the mapping when there is a perfect match. If any discrepancy happens, we consider the mapping to be imprecise and don't use it.

170 node ids out of 477 have a mapping of routes. It is observed that most of the time each vehicle runs one route, sometimes multiple vehicles operate the same route, and sometimes one vehicle operates multiple routes. As in Figure 4, node id 628 runs two routes 906 and 901. The trajectory can be matched from routes shown on STCP website.

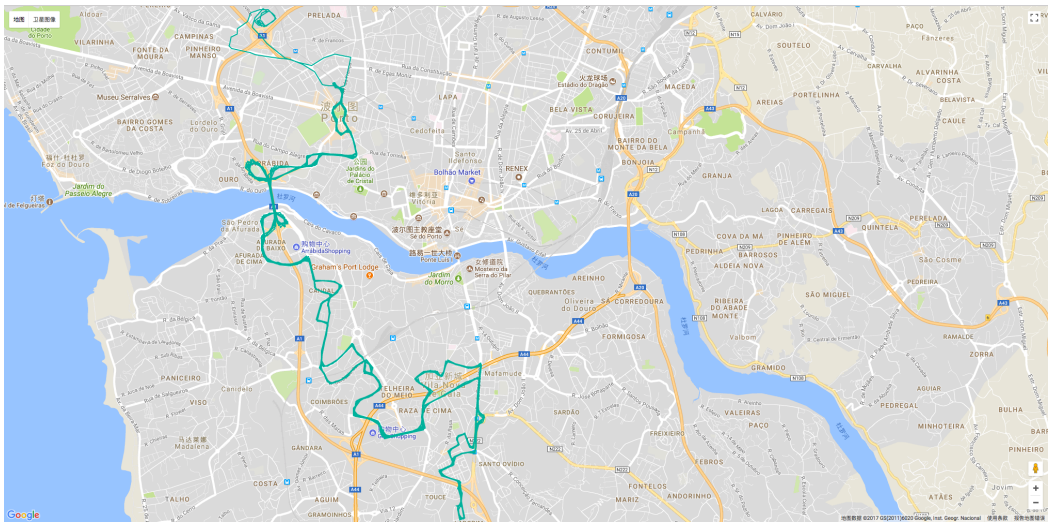
6 Methods

We develop three models to predict the bus travel time for routes and segment of routes. **Segment of route** is an arbitrary part of route that starts with one bus stop and ends with another.

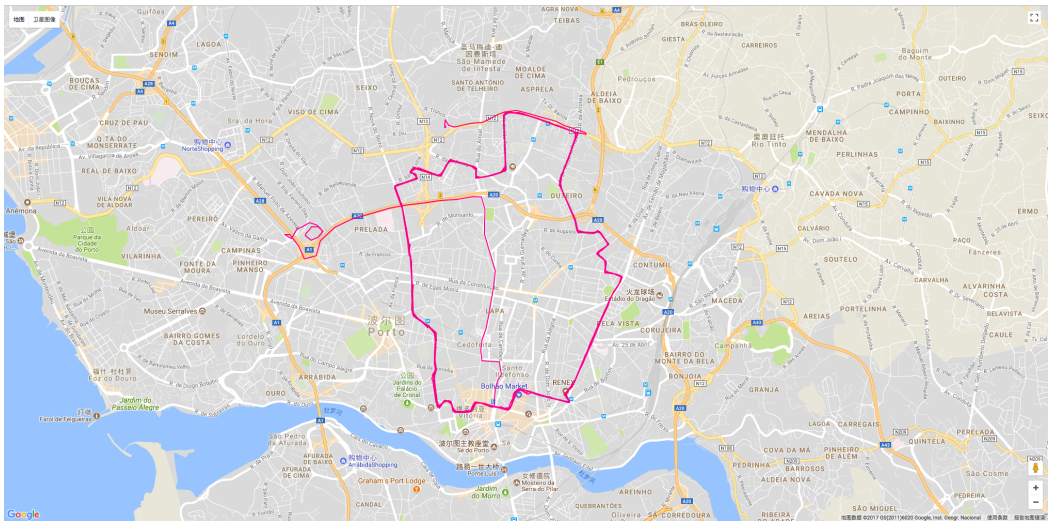
- **Whole route travel time prediction with GPS data**
  This model tries to predict the route travel time using bus GPS data and route prior knowledge. We tried three different sets of input features, from features computed purely from GPS data to features computed by GPS data integrating with prior knowledge. We see an increase in the accuracy as we embedded more prior knowledge to model input features.
- **Segment of route travel time prediction with multiple routes**
  This model predicts the segment travel time by using travel times of other buses running on the same segments. Preceding buses provide implicit traffic information, which helps with prediction accuracy.
- **Whole route travel time prediction with aggregation of segments**
  The third model predicts whole route travel time by adding travel times of continuous and non-overlapping segments along the route. Each segment travel time is predicted by the previous model.

In the next few section, we will first list the metrics used to compare the model performance. Then go over each model in detail, showing each model's purpose, overview, notation setup, computed features, data experiment and test results.

(a) Bus waits at bus gathering place before service, in left upper part of the map.

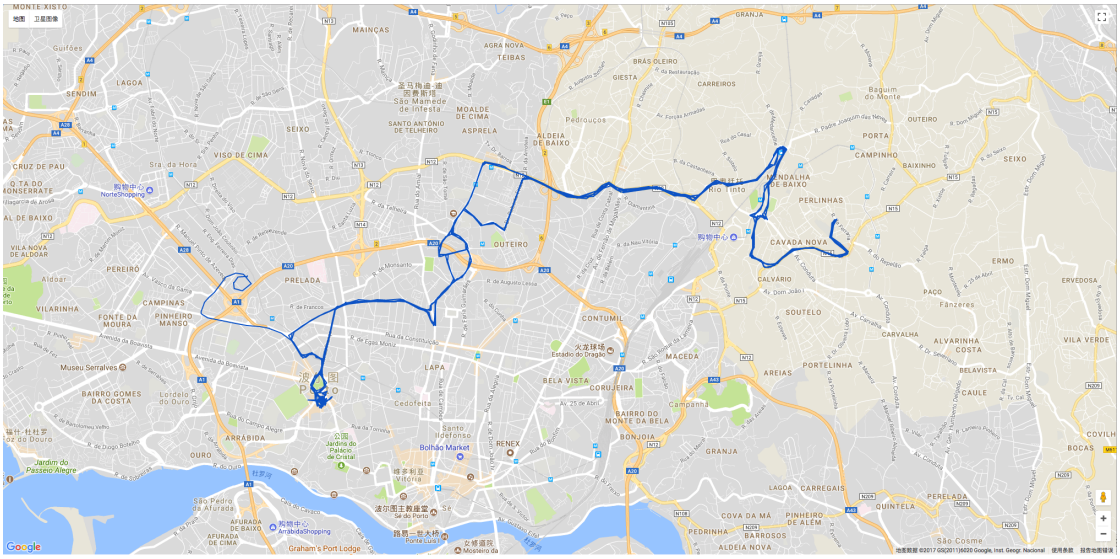


(b) Example of non-circular bus route

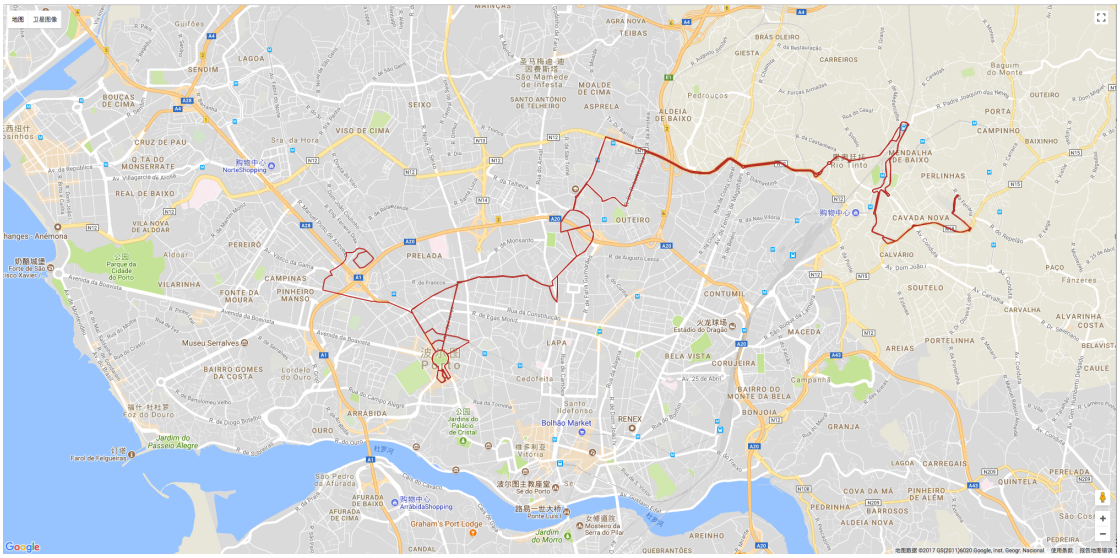


(c) Example of circular bus route

Figure 2: Trajectory for buses that are not moving, bus with non-circular route and with circular route.



(a) Before snapping to road, the thread is thicker and trajectory at traffic intersection does not lie on the road.



(b) After snapping, the polyline is tighter and the route is clearer.

Figure 3: Bus 114 trajectory before and after snapping.

6.1 Metrics

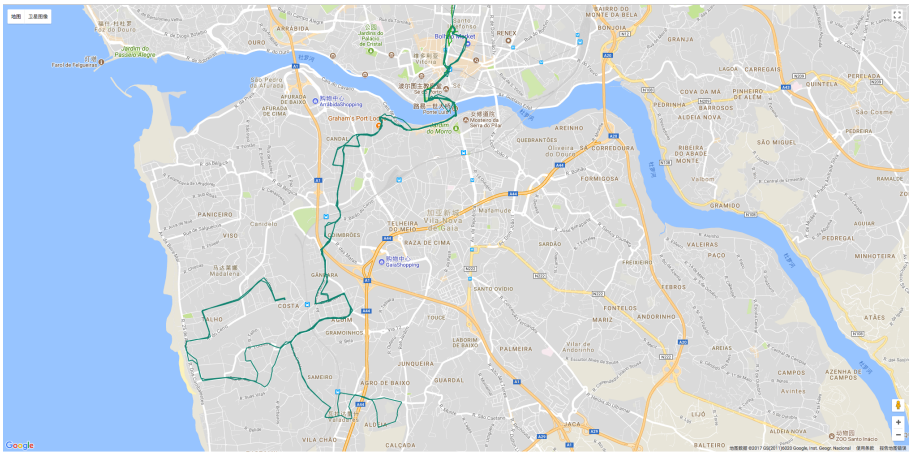
Suppose there are N test data points. Error between the predicted and actual travel time for the i-th sample is represented by $\epsilon_i = \hat{y}_i - y_i$. The metrics we use to measure the model performances are:

- **Mean Bias Error** measures the average difference between predict travel time and real travel time. Positive error and negative error will cancel out.

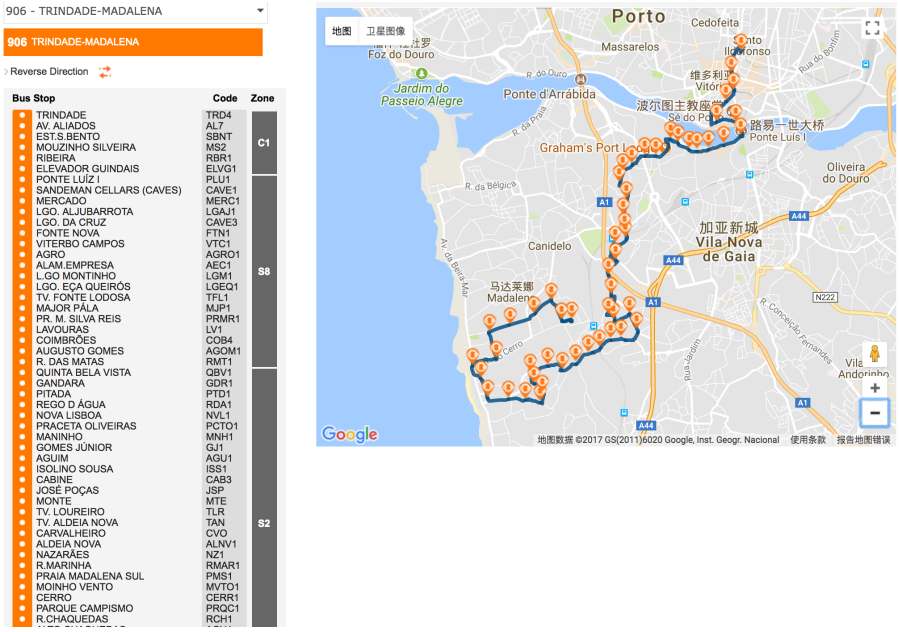
$$\textbf{MBE} = \frac{1}{N}\sum_{i=1}^{N}\epsilon_i$$

- **Maximum Absolute Error** measures the robustness: maximum deviation of prediction from real travel time.

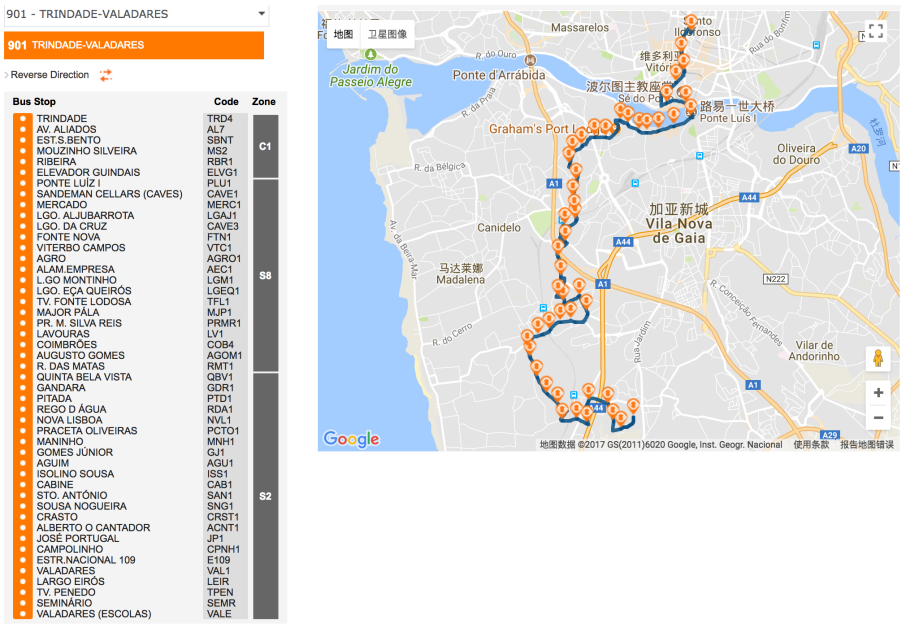
$$\textbf{MaxAE} = \max_{i\in 1:N}|\epsilon_i|$$



(a) Trajectory of node id 628



(b) Route 906 shown on website with sequence of bus stops



(c) Route 901 shown on website with sequence of bus stops

Figure 4: Node id 628 runs both route 906 and route 901. Shape of the trajectory matches shapes of the two routes.

- **Mean Absolute Error** measures the average derivation of prediction from real travel time.

$$\textbf{MeanAE} = \frac{1}{N}\sum_{i=1}^{N}|\epsilon_i|$$

- **Mean Absolute Percentage Error** measures the average percentage derivation from real travel time.

$$\textbf{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\epsilon_i}{y_i}\right| * 100$$

- **Root Mean Squared Error** measures average derivation. Compared to Mean Absolute Error, RMSE puts large weights on large errors. For this project, RMSE is preferable to MeanAE because large derivation is particularly undesirable when making travel time predictions, as it will drive bus riders away and make them miss the bus.

$$\textbf{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\epsilon_i^2}$$

- **Root Sum Squared Error** measures the overall deviation from predicted to actual travel time.

$$\textbf{RSSE} = \sqrt{\sum_{i=1}^{N}\epsilon_i^2}$$

6.2 Whole Route Travel Time Prediction with GPS Data

There are many factors that will influence the bus route travel time, such as traffic, weather, dwelling size... Intuitively, the more the factors are captured, the more accurate the predictions will be. So three models are developed sequentially, with same architecture but different set of input features. The first feature space is purely extracted from bus GPS data. For the second and third model, more and more prior knowledge is embedded. The three feature spaces are:

- **GPS data generated start point of route**
  Without prior knowledge, the start point of a route is chosen by the most visited and longest waited GPS position in a trajectory. In this model, we predict the travel time for the part between any two consecutive start points.
- **Start and end stops provided by route prior knowledge data**
  With the route prior knowledge, start and end stop GPS for a route can be obtained. We use this information to track each route. Generated input features are less noisy compared to the previous feature space.
- **Zones distribution provided by route prior knowledge data**
  Besides route start and end stops, the third feature space also uses zone information. As shown in the Figure 1, the zones bear different traffic weights. Inclusion of zone information implicitly bring bus route density factor to the model.

6.2.1 GPS data generated route start point

Model Overview
Suppose there is a target bus vehicle n, which operates on route l. We want to predict how long it takes for n to complete a route, using the route start point A_l, route start time $T_n^{A_l}$ and route length U_l. The numerical value of latitude and longitude at route start point is not important for the neural network, so we transform the (latitude, longitude) pair into one-hot categorical vectors $\vec{a}_l$, which determines how far A_l is from the center of downtown Porto, Portugal. In general, we expect the further route start point is, the less traffic, thus given same route length, the bus n will need less time. Route start time $T_n^{A_l}$ denotes the timestamp at which bus n arrives at route start point A_l. It is also transformed into an one-hot vector, $\vec{t}_n^{A_l}$. The bus starting during rush hours is expected to have longer travel time. Route length U_l is a positive real value, corresponding to the distance traveled during one route traversal. If other factors are the same, longer route costs more time.

The model makes the whole route travel time prediction by:

$$\hat{y}_{l,n} = f\left(\vec{a}_l, \vec{t}_n^{A_l}, U_l\right)$$

Computed Features
The three features are transformed as follows:

- $\vec{a}_l$ = one-hot(distance(A_l, p))
  We located the center of downtown Porto, denoted as p, at (41.155265, -8.61335) on Google Map. The one-hot vector has 5 ranges, corresponding to distance [$\leq$ 3km, 3km-5km, 5km-7km, 7km-10km, $\geq$ 10km].
- $\vec{t}_n^{A_l}$ = one-hot($T_n^{A_l}$)
  Time is also transformed to one-hot vectors, depending on which range the hour falls into: [1am-5am, 6am-11am, 11am-4pm, 4pm-9pm, 9pm-1am].
- U_l
  The distance the bus traveled between two consecutive route start points.

Data Processing

Algorithm 1 Generation of the feature space from bus GPS data

for *each trajectory s in a set of trajectories of bus n: S_n* **do**
  group GPS points within 150 meters for s
  fm_s = generate frequency map(s)
  A_l = sort(fm_s)[0]
  initialize $U_l = 0$
  initialize last time the bus passes route start point $T_n'^{A_l}$ = null
  for *each GPS and timestamp tuple (s_i^{lat}, s_i^{lng}, s_i^{ts}) in trajectory s* **do**
    U_l+ = *distance*$((s_i^{lat}, s_i^{lng}), (s_{i-1}^{lat}, s_{i-1}^{lng}))$ if $i \geq 1$
    if *(s_i^{lat}, s_i^{lng}) == A_l* **then**
      record vectorized A_l as $\vec{a}_l$;
      record vectorized s_i^{ts} as $\vec{t}_n^{A_l}$
      record U_l
      record $y = T_n^{A_l} - T_n'^{A_l}$ if $T_n'^{A_l}$ is not null
      update $U_l = 0$
      update $T_n'^{A_l} = T_n^{A_l}$
    end
  end
end

First we clean each trajectory by group GPS points within 150 meters. Then the cleaned trajectory is converted to a frequency map: with each unique GPS point as key and list of seconds the bus stays on the GPS point as value. For example, if a bus passes a point twice - the first time the bus stays 20 seconds, the second time 60 seconds, then the value should be (20, 60). We rank entries in the map first by how many times the bus stays at the GPS point, then by how much total time the bus stays on the point. The top entry is chose as the bus route start point for this trajectory.

After finding the route start point, we loop through the sequence of GPS points. When the bus passes the route start point, the timestamp is recorded and the distance from last time the bus passed the same point is taken as the route length. Then we convert route start point and timestamp to vector, and take the tuple as one input data point.

6.2.2 Start and end stops provided by route data

GPS data generated route start point itself is error-prone. This error will be amplified as we use wrong start point to find the route and generate the input features. Besides, without the end point of route, the travel time predicted is actually for routes traversed in both directions. In this model, we use prior knowledge of route start and end stops from Porto bus transportation website.

Model Overview
Suppose we have a target bus vehicle n, which operates on a route l that has a start point GPS A_l, and end point GPS B_l. We want to have a model that can predict the route travel time $\hat{y}_{l,n}$ for n from A_l to B_l, starting at timestamp $T_n^{A_l}$. A_l and B_l will be converted to vector $\vec{a}_l$, $\vec{b}_l$, representing the distance to the center of downtown Porto. Start time $T_n^{A_l}$ will also be converted to vector $\vec{t}_n^{A_l}$ to distinguish between regular hours and rush hours. The last input feature is the distance traveled for current route U_l.

The model will predict route travel time by:

$$\hat{y}_{l,n} = f\left(\vec{a}_l, \vec{b}_l, \vec{t}_n^{A_l}, U_l\right)$$

Computed Features

- $\vec{a}_l$ = one-hot(distance(A_l, p))
- $\vec{b}_l$ = one-hot(distance(B_l, p))
- $\vec{t}_n^{A_l}$ = one-hot$(T_n^{A_l})$
- U_l

Route start stop and end stop vectorization is the same as in the previous model. Calculate the distance between terminal stop and center of Downtown Porto p at (41.155265, -8.61335). Then categorize the distance according to [$\leq$ 3km, 3km-5km, 5km-7km, 7km-10km, $\geq$ 10km]. Categorize hour of time according to [1am-5am, 6am-11am, 11am-4pm, 4pm-9pm, 9pm-1am].

Data Processing
From bus GPS data, the input features are generated using algorithm 2.

First, we use actual bus stop GPS to replace trajectory GPS points if their distance is less than 45 meters. Then we can find out how long the bus spends on each stop. In real life, bus drivers usually rest a while at terminal stops, so that terminal stops have longer stay time than non-terminal stops. Thus we pick out stops in a trajectory that the bus stays for longer than 4 minutes and treat as a terminal stop. In the data experiment, we find clear route patterns using this method.

With the start and end stops chosen, GPS points in the trajectory are looped through. We extract the path from start stop to end stop and vice versa. Then record the start stop GPS, end stop GPS, start stop arrival time, path distance, and travel time as one input data point.

6.2.3 Zones distribution provided by route data

The actual travel time is hugely influenced by the traffic along the path from start stop to end stop. So we use zone information to capture the traffic. For each trajectory, we partition GPS into zones. Determining how many recorded data points in each zone roughly tells the traffic combinations for the path, thus making the prediction more accurate.

Model Setup
Suppose there is a target bus n, running on route l, for which we want to predict the route travel time $\hat{y}_{l,n}$. l starts at bus stop A_l and ends at bus stop B_l. Considering a route traversal from A_l to B_l or vice versa, represented as a sequence of (latitude, longitude, timestamp) tuples. $Z_{l,n}$ is the distribution of how many GPS points in a route traversal lies in each zone. There are 18 zones: [C1, C2, C3, C4, C5, C6, C8, C9, C10, C11, C16, S1, S2, S8, S9, N10, N11, N16]. So $Z_{l,n}$ is a 18x1 vector. Time-wise, we transfer the timestamps to a vector showing how many timestamps fall in each time range. The time distribution $W_{l,n}$ is a 5x1 vector with ranges: [hour 1-5, hour 6-11, hour 11-16, hour 16-21, hour 21-1]. The third feature is the length of route U_l.

Travel time for the whole route is predicted by:

$$\hat{y}_{l,n} = f(Z_{l,n}, W_{l,n}, U_l).$$

Computed Features
We use 1-nearest-neighbors to assign the zone number of the closest bus stop to the GPS point.

Algorithm 2 Generation of input feature using GPS data and route prior knowledge

for *each trajectory s in a set of trajectories for bus n:* S_n **do**
  for *each tuple* (s_i^{lat}, s_i^{lng}, s_i^{ts}) *in trajectory s* **do**
    for *each bus stop m with GPS* (m^{lat}, m^{lng}) *in the set of bus stops:* M **do**
      if *distance*((s_i^{lat}, s_i^{lng}), (m^{lat}, m^{lng})) $\leq$ *45m* **then**
        (s_i^{lat}, s_i^{lng}) = (m^{lat}, m^{lng})
      end
    end
  end
  initialize A_l, B_l, $T_n^{A_l}$ = null
  initialize $U_l = 0$
  for *each GPS and timestamp tuple* (s_i^{lat}, s_i^{lng}, s_i^{ts}) *in trajectory s* **do**
    U_l += distance((s_i^{lat}, s_i^{lng}), (s_{i-1}^{lat}, s_{i-1}^{lng})) if i $\geq$ 1
    if (s_i^{lat}, s_i^{lng}) *is in bus stops set:* M **then**
      T' = timestamp bus leaves (s_i^{lat}, s_i^{lng})
      if T' - $s_i^{ts} \geq$ *4 minutes* **then**
        if A_l == *null* **then**
          A_l = (s_i^{lat}, s_i^{lng})
          $T_n^{A_l} = s_i^{ts}$
        end
        else
          if (s_i^{lat}, s_i^{lng}) == A_l **then**
            update $T_n^{A_l} = s_i^{ts}$
            update $U_l = 0$
          end
          else
            B_l = (s_i^{lat}, s_i^{lng})
            record $\vec{a}_l$ = vectorized A_l, $\vec{b}_l$ = vectorized B_l, $\vec{t}_n^{A_l}$ = vectorized $T_n^{A_l}$, U_l, $y = s_i^{ts} - T_n^{A_l}$
            update $A_l = B_l$
            update $T_n^{A_l} = s_i^{ts}$
            update $U_l = 0$
            jump the loop to after bus leaves (s_i^{lat}, s_i^{lng})
          end
        end
      end
    end
  end
end

- $Z_{l,n} = \left[\sum_i I(zone(s_i^{lat}, s_i^{lng}) = C1), ..., \sum_i I(zone(s_i^{lat}, s_i^{lng}) = N16)\right]$
- $W_{l,n} = [\sum_i I(s_i^{ts} \in hour(1,5)), ..., \sum_i I(s_i^{ts} \in hour(21,1))]$
- U_l

Data Processing
The algorithm 3 shows how to generate input features for this model.

There are three steps to transfer the data to the computed features. First, we snap the GPS point to bus stop GPS within 45 meters. Then the same method to find start and end stops as in the previous model is used. After getting the start and end stops, (lat, lng, timestamp) tuples for the trajectory are looped through. Once a terminal bus stop is reached, the algorithm starts to aggregate zone and time distribution of each GPS point encountered until the other terminal stop is reached. Then record the computed features, re-initialize the features and repeat the same collecting process, until whole trajectory is looped through.

Algorithm 3 Generation of input feature using bus GPS data and zone, route prior knowledge

for *each trajectory s in a set of trajectories for bus n:* S_n **do**
  for *each tuple* (s_i^{lat}, s_i^{lng}, s_i^{ts}) *in trajectory s* **do**
    for *each bus stop m with GPS* (m^{lat}, m^{lng}) *in the set of bus stops: M* **do**
      if *distance*((s_i^{lat}, s_i^{lng}), (m^{lat}, m^{lng})) $\leq$ *45m* **then**
        (s_i^{lat}, s_i^{lng}) = (m^{lat}, m^{lng})
      end
    end
  end
  initialize $Z_{l,n}$ = [0] * 18
  initialize $W_{l,n}$ = [0] * 5
  initialize U_l = 0
  for *each GPS and timestamp tuple* (s_i^{lat}, s_i^{lng}, s_i^{ts}) *in trajectory s* **do**
    m' = KNN((s_i^{lat}, s_i^{lng}), M, k=1)
    $Z_{l,n}$ += vectorize (zone(m'))
    $W_{l,n}$ += vectorize (s_i^{ts})
    U_l += distance((s_i^{lat}, s_i^{lng}), (s_{i-1}^{lat}, s_{i-1}^{lng})) if $i \geq 1$
    if (s_i^{lat}, s_i^{lng}) *is in bus stops set: M* **then**
      T' = timestamp bus leaves (s_i^{lat}, s_i^{lng})
      if T' - $s_i^{ts} \geq$ *4 minutes* **then**
        if A_l == *null* **then**
          A_l = (s_i^{lat}, s_i^{lng})
          $T_n^{A_l} = s_i^{ts}$
          update $Z_{l,n}$ = [0] * 18
          update $W_{l,n}$ = [0] * 5
          update U_l = 0
        end
        else
          if (s_i^{lat}, s_i^{lng}) == A_l **then**
            update $T_n^{A_l} = s_i^{ts}$
            update $Z_{l,n}$ = [0] * 18
            update $W_{l,n}$ = [0] * 5
            update U_l = 0
          end
          else
            record $Z_{l,n}$, $W_{l,n}$, U_l, $y = s_i^{ts} - T_n^{A_l}$
            update A_l = (s_i^{lat}, s_i^{lng})
            update $T_n^{A_l} = s_i^{ts}$
            update $Z_{l,n}$ = [0] * 18
            update $W_{l,n}$ = [0] * 5
            update U_l = 0
            jump the loop to after bus leaves (s_i^{lat}, s_i^{lng})
          end
        end
      end
    end
  end
end

6.2.4 Learning Model

All three sets of features are fed to three-layer neural networks. The first model takes three computed features $\vec{a}_l$, $\vec{t}_n^{A_l}$, U_l, feeding to network with 11 input nodes, 6 hidden units, and 1 output unit. The second model takes four computed

features $\vec{a}_l$, $\vec{b}_l$, $\vec{t}^{A_l}_n$, U_l, feeding to network with 16 input nodes, 11 hidden units, and 1 output unit. The third model takes three computer features $Z_{l,n}$, $W_{l,n}$, U_l, feeding to a network with 24 input units, 13 hidden units and 1 output unit.

Bias are included in hidden and output layers. Hidden units are activated by tanh and output layer is linear function. All three models try to minimize the loss function given by Mean Squared Error:

$$Loss = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2.$$

Input dataset is divided into training set (70%) and test set (30%). 20% of test set are used as the validation set. Training set is shuffled and 16 data points forming a batch. We use Adam optimizer to train the model parameters, with adaptive learning rate starting at 0.001. If there is no improvement in validation loss for 10 epochs, decrease the learning rate by 0.1. The model stops training until learning rate reaches 1e-8.

6.2.5 Model Input Data Properties

Model 2 and 3 uses the same method to get start and end stops, so they have same path partition data. Because model 1 uses only start bus stop while the other two models use both start and end bus stops, the mean value of y for model 1 is larger than the mean value of y in the other two models.

Model	# of samples	Min (m)	Max (m)	Mean (m)	Std
model 1	1041	21	197	54.67	23.83
model 2&3	932	10	232	42.45	24.85

Table 3: Statistics for model 1 input.

6.2.6 Results

All three models have small Mean Bias Error, however this does not mean the model is making good prediction, because the positive error and negative error can cancel out. The Mean Absolute Error tells that on average model 1 predicts the route travel time 23 minutes away, 12 minutes for model 2 and 14 minutes for model 3. All models make large prediction error on some routes. These large route prediction errors result in larger Root Mean Squared Error than their respective Mean Absolute Error.

Comparing cross models, we do see a drop in Mean Absolute Percentage Error, as embedding more prior knowledge to input features. This finding is inspiring because our ultimate goal is to absorb as many factors as possible to make good travel time prediction.

Model	mean of y (m)	std of y	MBE	MaxAE	MeanAE	MAPE	RMSE	RSSE
model 1	54.6702	23.6525	-13.5874	96.9902	22.5471	43.1676%	28.8501	351.82209
model 2	42.3944	24.2462	-3.6364	116.3902	11.8916	27.7715%	21.5533	207.4756
model 3	42.4539	24.5109	-10.2638	124.4084	13.754	26.6366%	24.1624	233.1805

Table 4: Result for route travel time prediction model using bus GPS data.

6.3 Segment Travel Time Prediction Model with Multiple Routes

Besides making prediction on whole route, it is also useful to make prediction on segment of the route. Segment of route is an arbitrary part of the route that between any two bus stops. Given a bus is at stop A, the model can predict how long it takes for the bus to go from A to stop B, thus providing bus riders at stop B with bus arrival time prediction. Bus arrival information is particularly important because it saves people time waiting at bus stops and allow them to make better time planning.

This model solves the problem of bus travel time prediction using multiple routes. Multiple routes provide more references while making predictions, and also improve data timeliness and reliability, thus increasing model performances.

6.3.1 Model Overview

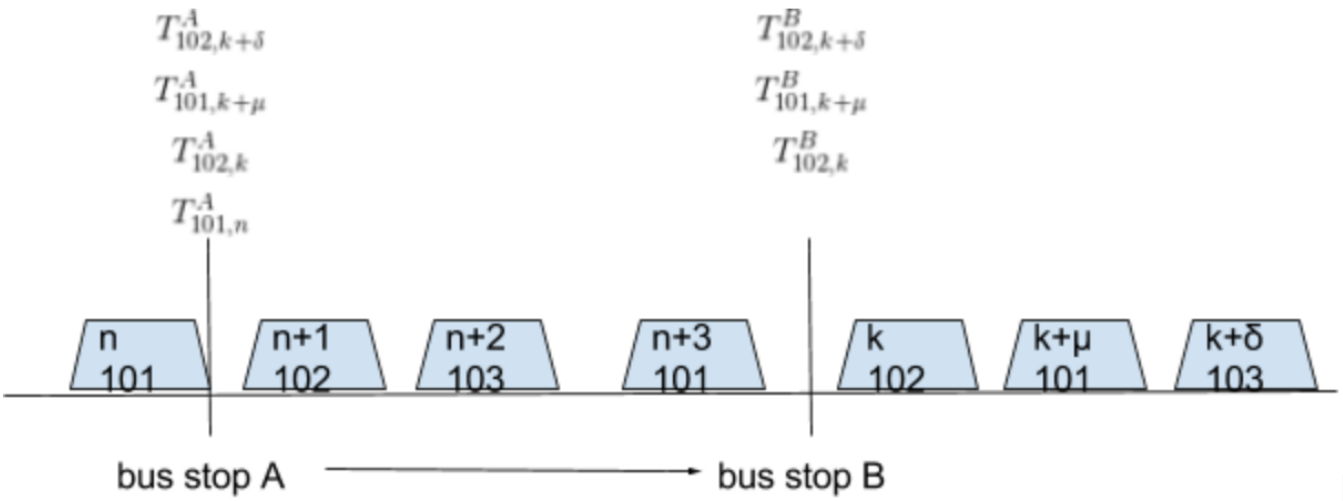


Figure 5: Illustration of preceding buses.

Suppose that there are three routes 101, 102, 103, that runs on one common segment: from bus stop A to bus stop B. We want to predict the segment travel time for a target bus n. Bus n operates route 101. When bus n arrives at bus stop A at timestamp $T^A_{101,n}$, there are bus $n+1$ (route 102), $n+2$ (route 103), $n+3$ (route 101)... that are between bus stop A and bus stop B. There are also bus k (route 102)...$k+\mu$ (route 101)...$k+\delta$ (route 103) that proceed from bus stop A to bus stop B and pass B. We treat these buses that have passed the end bus stop as **preceding buses**. This is illustrated in Figure 5.

We use the travel time of preceding buses on segment A to B to predict the travel time for target bus n. Traffic is highly time sensitive, we expect traffic ten minutes ago is more like current traffic than traffic one hour ago. So travel time of preceding buses will make a more accurate prediction than that of a non-preceding bus. The time closeness between preceding bus of any routes and the target bus n is denoted as $t^C_{L,n}$ ('C' means closeness), where L is set of routes that pass segment A-B. The smaller the $t^C_{L,n}$ is, the more accurately preceding bus reflects current traffic situation. We use the weighted average travel time of δ preceding buses as another input feature $\bar{t}^r_{L,n}$ ('r' means running time). The weight is given by the normalized inverse time closeness, which gives more weight if the bus is timely closer to the target bus. Preceding bus that operates on same route bears more similarities to target bus. Therefore another input feature is the running time of the preceding bus of the same route l, denoted as $t^r_{l,n}$. The time closeness between this bus and target bus is taken as $t^c_{l,n}$.

The model predicts the target bus travel time by these four computed features:

$$\hat{y}_{l,n} = f(t^C_{L,n}, t^c_{l,n}, \bar{t}^r_{L,n}, t^r_{l,n})$$

6.3.2 Computed Features

We want to predict the travel time $\hat{y}_{l,n}$ of target bus n, which operates on route l, for a pre-specified segment from bus stop A to B. L is the set of routes that pass this segment.

- $t^C_{L,n} = T^A_{l,n} - T^A_{L,k}$ time closeness between target bus n and the timely closest preceding bus k of any route in L, measured by the arrival time difference at bus stop A for bus n and preceding bus k.

- $t^c_{l,n} = T^A_{l,n} - T^A_{l,k+\mu}$ time closeness between target bus n and preceding bus $k+\mu$ of same route l, measured by the arrival time difference at bus stop A for bus n and $k+\mu$.
- $t^r_{l,n} = T^B_{l,k+\mu} - T^A_{l,k+\mu}$ segment travel time of preceding bus $k+\mu$ of same route l, measured by the arrival time difference between bus stop A and bus stop B.
- $\bar{t}^r_{L,n} = \sum_{j=1}^{\delta} \frac{1/\left(T^A_{l,n} - T^A_{L,k+j}\right)}{\Gamma} \left(t^r_{L,k+j}\right)$,
  where $\Gamma = \sum_{j=1}^{\delta} 1/\left(T^A_{l,n} - T^A_{L,k+j}\right)$
  $\bar{t}^r_{L,n}$ is the weighted average segment travel time of δ preceding bus $k, ...k+\delta$ of any routes in the route set L. Travel time for each preceding bus is measured by arrival time difference between bus stop A and bus stop B. The weight is the normalized inverse of the time closeness between the preceding bus and the target bus.

6.3.3 Model Setup

We feed the four computed features $t^C_{L,n}, t^c_{l,n}, \bar{t}^r_{L,n}, t^r_{l,n}$ to a three-layer neural network. The network has 4 input nodes, 5 hidden units and 1 output units. Bias included in hidden and output layers. Hidden units are activated by tanh. Output activation is linear, given travel time prediction is real-valued. The model tries to minimize the loss function given by Mean Squared Error:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 .$$

We use Adam optimizer to train the weights, with adaptive learning rate starting at 0.1.

6.3.4 Data Experiment

We use small bus GPS dataset to do the data experiment. First we picked four segments, each goes one of the four directions: south, north, west, east. Then we process the data to compute features for the network. After the computed features are generated, we partition the computed features data into training and test set. Then we train and test the network and analyze the results.

Segments Selection

The four segments are shown in table below. They cover four directions and are located in different zones. Also they are passed by many bus routes, thus we can collect reasonably large amount of data. More data leads to better timeliness and reliability of the computer features. The relative positions of the four segments can be seen in Figure 6. And statistics for each segment is shown in table 5.

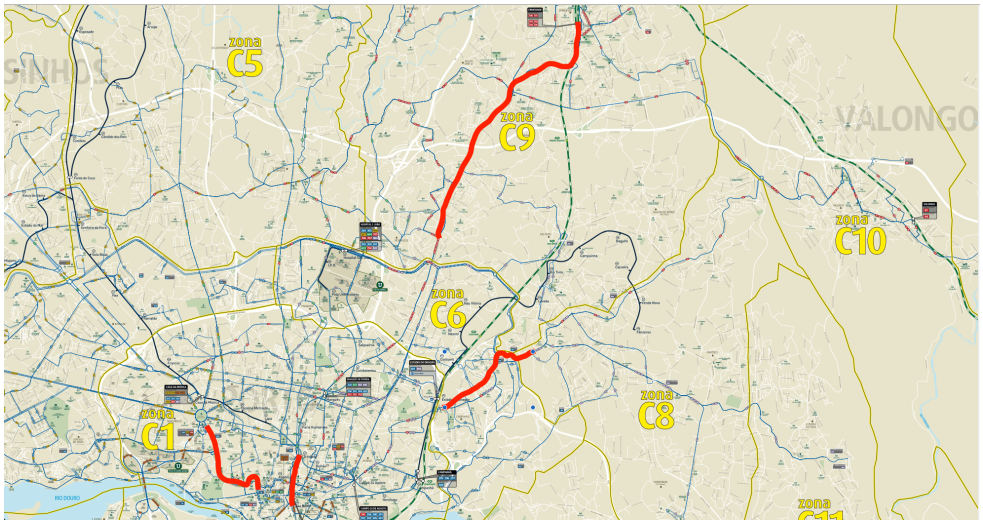


Figure 6: Relative position of four segments on map.

Direction	From bus stop - To bus stop	Routes
North	AREOSA MERCADO (ARSM1) - ERMESINDE FORUM (ERM4)	701, 703, 704, 705, 707
South	TRINDADE (TRD3) - SAO BENTO (SBNT)	200, 202, 305, 900, 901, 904, 905, 906
East	CORUJEIRA (CRJ1) - TV. DA PONTE (TPT1)	700, 700_V94, 800, 801, 806
West	CARMO (CM0) - BOAVISTA B. BUCESSO (BS1)	201, 208, 302, 507, 601

Table 5: Four segment selection: start and end bus stops and routes passing the segment.

Algorithm 4 Finding Arrival Time for Segment Starting and Ending Stops

for *each segment* d **do**
  Get start and end bus stop GPS (A_d, B_d) from bus stops prior knowledge
  for *each route* l *in set of routes that pass* d*:* L_d **do**
    Find node ids that operates route l from node id to route mapping
    for *each node id* n *that runs route* l **do**
      for *each trajectory* s *in set of trajectories for bus* n*:* S_n **do**
        for *each GPS and timestamp pair* (s_i^{lat}, s_i^{lng}, $s_i^{timestamp}$) *in trajectory* s **do**
          if $vincenty\Big((s_i^{lat}, s_i^{lng}), A_d\Big) \leq 50m$ **then**
            record arrival time $T_{l,n}^{A_d} = s_i^{timestamp}$ for segment d
          end
          if $vincenty\Big((s_i^{lat}, s_i^{lng}), B_d\Big) \leq 50m$ **then**
            record arrival time $T_{l,n}^{B_d} = s_i^{timestamp}$ for segment d
          end
        end
      end
    end
  end
end

Algorithm 5 Computing input features from bus stop arrival time

for *each segment* d **do**
  get start and end stop GPS pair (A_d, B_d) from prior knowledge
  for *each bus stop arrival time pairs* ($T_{l,n}^{A_d}$, $T_{l,n}^{B_d}$) *for segment* d **do**
    Calculate:
    $t_{L,n}^{C} = T_{l,n}^{A_d} - T_{L,k}^{A_d}$ if k is present
    $t_{l,n}^{c} = T_{l,n}^{A_d} - T_{l,k+\mu}^{A_d}$ if $k+\mu$ is present
    $t_{l,n}^{r} = T_{l,k+\mu}^{B_d} - T_{l,k+\mu}^{A_d}$ if $k+\mu$ is present
    $\bar{t}_{L,n}^{r} = \sum_{j=1}^{3} \frac{1/\left(T_{l,n}^{A_d} - T_{L,k+j}^{A_d}\right)}{\sum_{j=1}^{3} 1/\left(T_{l,n}^{A_d} - T_{L,k+j}^{A_d}\right)} \left(T_{L,k+j}^{B_d} - T_{L,k+j}^{A_d}\right)$ if there are 3 preceding buses
    $y = T_{l,n}^{B_d} - T_{l,n}^{A_d}$
    if *all four features exist* **then**
      record $\left(t_{L,n}^{C}, t_{l,n}^{c}, \bar{t}_{L,n}^{r}, t_{l,n}^{r}, y\right)$ for segment d
    end
  end
end

Data Processing

The procedure to process raw data to get bus arrival timestamps at stops for each segment for each bus vehicle is shown in algorithm 4. The next step is to compute input features from timestamp pairs $(T^A_{l,n}, T^B_{l,n})$, as shown in algorithm 5. There are three steps to generate model input features from the bus GPS raw data. First, since we know the routes that transit through each segment and the mapping from node id to route, we can find out the bus vehicles that pass each segment. Then we loop through the trajectory and record bus arrival time at start and end stops of each segment. We consider GPS within 50 meters of the stop as "passing" the stop. Then we compute the four features using the arrival timestamps.

Data Experiment
North segment is the longest: 4.44km, but with lightest traffic. Long distance makes the travel time have high variance. South segment is the shortest but traffic-heaviest, which also implies high variance. On the other hand, east and west segments have relatively stable travel time.

Direction	From - To	# of samples	Distance (Approx.)	Min Travel Time (m)	Max (m)	Mean (m)	Std
North	ARSM1 - ERM4	36	4.44km	8	53	21.23	13.150
South	TRD3 - SBNT	67	0.63km	2	65	11.915	9.007
East	CRJ1 - TPT1	60	1.65km	4	8	5.204	0.802
West	CMO - BS1	58	1.39km	4	8	5.698	0.902

Table 6: Statistics for four segments picked.

Learning Model
We have four group of data. We performed predictions on each group as well as on four groups combined. In both situations, 70% of the data are used as training set, 30% test set. We set batch size to be 10 for training set and shuffle the data. During training, the model starts to overfit around 200 epochs.

Results
In general, the model has little mean bias error. Noisy inputs lead to high error rate. Clean data in the east and west direction have high prediction accuracy, with Mean Absolute Error of 0.8 and 0.7 minutes. North and south segments have high Mean Absolute Error, which means the prediction on average deviates 9.1 and 4.6 minutes away from the actual travel time. For south, east ans west segments, the Mean Absolute Error is closer to Root Mean Squared Error, while north segment makes large prediction errors on some data points. This also proves that north segment's Max Absolute Error, 25.8 minutes, is far larger than the counterparts of other segments. After combining all four directions together, error rate is around 55%. The model works really well with east and west segments, less well with four directions combined, worse with north and worst with south.

Model	mean of y (m)	std of y	MBE	MaxAE	MeanAE	MAPE	RMSE	RSSE
North	21.2285	12.8552	-7.6794	25.8103	9.0903	57.9887%	13.9004	28.7475
South	11.8293	7.4351	-1.4125	15.9499	4.6824	59.2737%	6.9571	20.3487
East	5.2015	0.7467	-0.3491	1.9029	0.8502	16.8405%	1.0292	2.8681
West	5.7066	0.8195	-0.0212	1.7414	0.7028	12.8016%	0.9134	2.5181
All Directions	9.7965	8.7411	-1.5753	29.3046	3.7999	55.2241%	7.6735	34.0372

Table 7: Results for the segment travel time prediction model.

6.4 Whole Route Travel Time Prediction with Combined Segments

This model partitions a route into segments and make segment travel time predictions by using preceding buses information. The final route travel time prediction is the addition of all segment predictions. The negative and positive error of segment predictions will cancel out, so we expect this route travel time prediction model to outperform previous three models.

6.4.1 Model Overview

On the Porto city bus transportation system, we pick a set of bus stop "hubs", where more routes pass these hubs than other bus stops. The set of hubs is denoted as H. Route between any two bus hubs is considered a segment. For a bus vehicle n, operating on route l, the timestamp n passes the hub h_i is denoted as $T_{l,n}^{h_i}$. Route l will be separated by hubs H into several segments, denoted as D_l. For each segment d in D_l, we use bus hubs arrival timestamp to calculate the four input features: $t_{L,n,d}^{C}$, $t_{l,n,d}^{c}$, $\bar{t}_{L,n,d}^{r}$, $t_{l,n,d}^{r}$. The model takes the four features and make segment travel time prediction. The final route travel time prediction is given by:

$$\hat{y}_{l,n} = \sum_{d}^{D_l} f\left(t_{L,n,d}^{C}, t_{l,n,d}^{c}, \bar{t}_{L,n,d}^{r}, t_{l,n,d}^{r}\right)$$

6.4.2 Computed Features

The model has the same four computed features as the segment travel time prediction model.

- $t_{L,n,d}^{C} = T_{l,n}^{A_d}$ - $T_{L,k}^{A_d}$
  For segment d, the start bus hub is denoted as A_d. k is the preceding bus for segment d. Time closeness is given by arrival time difference at segment start bus hub.
- $t_{l,n,d}^{c} = T_{l,n}^{A_d} - T_{l,k+\mu}^{A_d}$
  $k+\mu$ is the preceding bus that operates the same route l. Time closeness is given by arrival time difference at segment start bus hub.
- $t_{l,n,d}^{r} = T_{l,k+\mu}^{B_d} - T_{l,k+\mu}^{A_d}$
  Running time of the preceding bus of route l is the arrival time different between the two bus hubs.
- $\bar{t}_{L,n,d}^{r} = \sum_{j=1}^{\delta} \frac{1/\left(T_{l,n}^{A_d} - T_{L,k+j}^{A_d}\right)}{\Gamma} \left(t_{L,k+j,d}^{r}\right)$
  $\Gamma = \sum_{j=1}^{\delta} 1/\left(T_{l,n}^{A_d} - T_{L,k+j}^{A_d}\right)$
  Weighted average travel time of preceding buses on same segment d.

6.4.3 Model Setup

This model has the same setup as segment travel time prediction model.

6.4.4 Data Experiment

We conduct the experiment on the small bus GPS dataset and gather bus hub GPS from bus transportation website.
Bus stop hubs selection

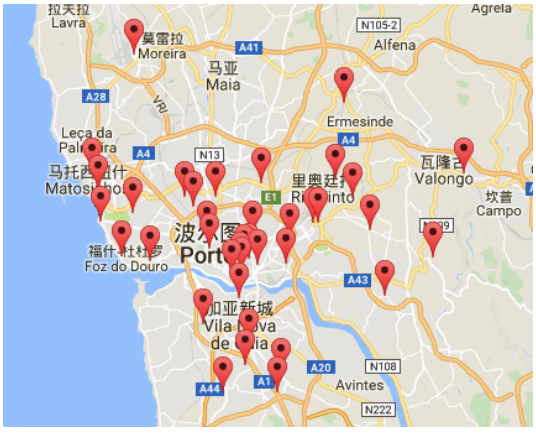


Figure 7: Stops picked for whole route prediction with combined segments.

36 bus hubs are selected, and their codes are: COB4, AEPT1, SRGB1, STLZ2, VIS3, SBNT2, GODS, BS6, RFAR1, MPL3, BLFZ1, VALE, SO3, VDE, ACV, AL7, LBM1, RTE4, RTES1, PT3, C24A5, MTSP1, CQ10, CMP1, ERMI2, MV3, HSJ10, CMS2, ALD1, EDRG1, TRD3, VLGE1, MATM1, SPC, GNR1, CORD4. They spread across zones, and considerable number of routes pass them. Their position on the map is shown in Figure 7.

Data processing

Algorithm 6 Generating segment traversal information

for *each bus* n **do**
  for *each trajectory* s *in set of trajectories for bus* n*:* S_n **do**
    initialize A_d = null
    initialize B_d = null
    for *each* $(s_i^{lat}, s_i^{lng}, s_i^{ts})$ *in trajectory* s **do**
      h' = closest bus hub $((s_i^{lat}, s_i^{lng})$, H)
      if *distance*$((s_i^{lat}, s_i^{lng}), h') \leq 150$ *meters* **then**
        if A_d == *null* **then**
          $A_d = h'$
          $T_{l,n}^{A_d} = s_i^{ts}$
        end
        else
          if $A_d == h'$ **then**
            $T_{l,n}^{A_d} = s_i^{ts}$
          end
          else
            $B_d = h'$
            $T_{l,n}^{B_d} = s_i^{ts}$
            record $T_{l,n}^{A_d}$, $T_{l,n}^{B_d}$, n, l to file "$A_d - B_d$.txt"
            update $A_d = B_d$, $T_{l,n}^{A_d} = T_{l,n}^{B_d}$
          end
        end
      end
    end
  end
end

There are two steps to generate the input data from bus GPS data. First is to construct segment traversal information for all buses, i.e. get bus hubs arrival timestamps from each bus (algorithm 6). The second step is to partition trajectories into route traversals and partition each traversal into segments and compute input features (algorithm 7).

Result

We measure the model for both segment prediction and route prediction. 1030 route traversals and 3127 segment traversals input data were gathered. As shown in Figure 8, the model can make good predictions in most cases. The red line shows the prediction error and it swings around 0. Also, Mean Bias Error is low. The model does well at predicting outliers, given the fact that Mean Absolute Error is close to Root Mean Squared Error. Also this model effectively cancels out prediction errors among segments, resulting in lower MAPE for route predictions than for segment predictions, shown in table 8.

Model	# of samples	Min (m)	Max (m)	Mean (m)	Std
segment	3127	1	155	8.9936	10.4955
route	1030	1	185	27.3039	21.3237

Table 8: Statistics for model 3 input.

Algorithm 7 Generating model input features

for *each bus* n **do**
  for *each trajectory* s *in set of trajectories for bus* n*:* S_n **do**
    segments = []
    visitedHubs = []
    initialize A_d = null, B_d = null
    for *each* $(s_i^{lat}, s_i^{lng}, s_i^{ts})$ *in trajectory* s **do**
      h' = closest bus hub $((s_i^{lat}, s_i^{lng})$, H)
      if *distance*$((s_i^{lat}, s_i^{lng}), h') \leq 150$ *meters* **then**
        if A_d == *null* **then**
          $A_d = h'$
          $T_{l,n}^{A_d} = s_i^{ts}$
        end
        else
          if $A_d == h'$ **then**
            $T_{l,n}^{A_d} = s_i^{ts}$
          end
          else
            $B_d = h'$
            $T_{l,n}^{B_d} = s_i^{ts}$
            if h' *not in visitedHubs* **then**
              open file "$A_d - B_d$.txt" and compute $t_{L,n,d}^{C}$, $t_{l,n,d}^{c}$, $\bar{t}_{L,n,d}^{r}$, $t_{l,n,d}^{r}$
              compute $y = T_{l,n}^{B_d} - T_{l,n}^{A_d}$
              segments.append($[t_{L,n,d}^{C}$, $t_{l,n,d}^{c}$, $\bar{t}_{L,n,d}^{r}$, $t_{l,n,d}^{r}$, $y]$)
            end
            else
              record segments
              update segments = []
              update visitedHubs = $[B_d]$
            end
            update $A_d = B_d$, $T_{l,n}^{A_d} = T_{l,n}^{B_d}$
          end
        end
      end
    end
  end
end

Model	mean of y (m)	std of y	MBE	MaxAE	MeanAE	MAPE	RMSE	RSSE
segment	9.4636	10.3567	-0.4279	62.8879	2.2907	50.0264%	5.7146	101.0200
route	5.2446	21.1207	-1.2955	62.4300	5.2446	32.5633%	10.4480	106.0359

Table 9: Results for segment travel time prediction model.

7 Model Improvements

We made three modifications: use the larger dataset, predict travel time by time of the day and modify the loss function.

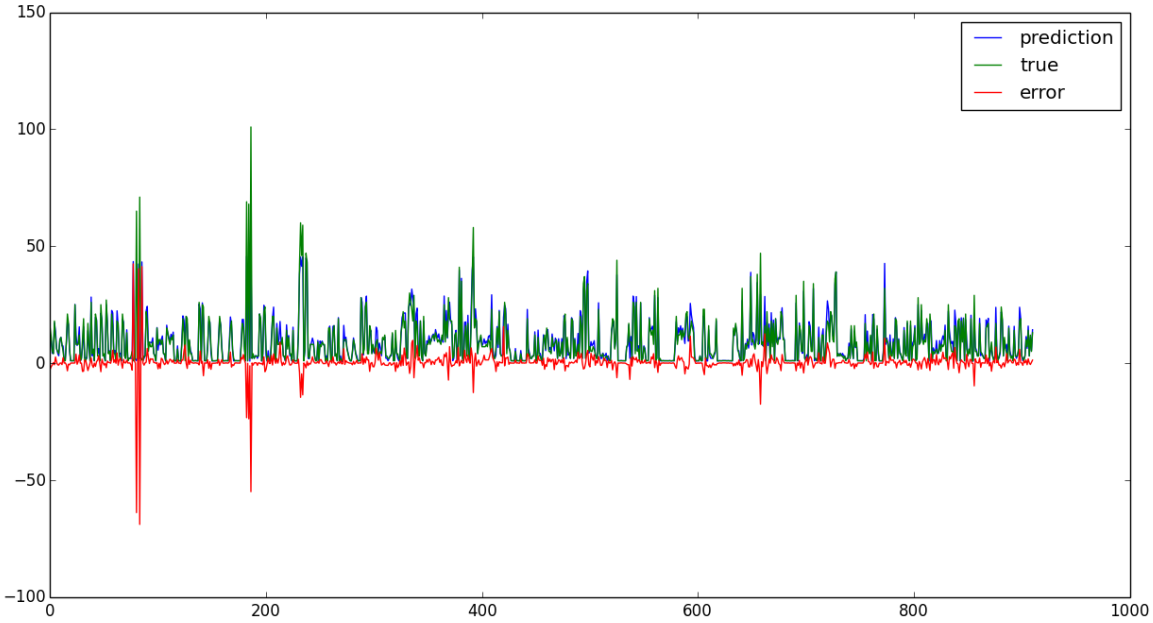


Figure 8: Prediction travel time for each segment, true travel time and their difference.

7.1 More Data

It is desirable if model has higher accuracy with more data. To test this, we use the large bus GPS dataset in the first model (route travel time prediction using bus GPS data generated starting points). Traffic on different day of the week differs a lot. So we only use Sunday's data from the large dataset, since data in the small dataset only covers Sunday (data for Monday and Tuesday contains only one bus vehicle and the vehicle is not moving). The large dataset provides 18 more data points. We experimented on using only 50% v.s. 100% of the data combining the small and large datasets. We got the following results. As the metrics show, larger dataset increases the Max Absolute Error, however improving the overall prediction accuracy.

Model	mean of y (m)	std of y	MBE	MaxAE	MeanAE	MAPE	RMSE	RSSE
50% data	56.1125	25.5773	-12.0337	93.8483	24.0119	44.6895%	30.8083	225.2527
100% data	54.8319	24.6186	-12.0790	108.2958	21.8783	40.5001%	29.1240	299.7227

Table 10: Metrics for route travel time prediction model using 50% of the data v.s. 100% of the data (small dataset plus Sunday data in the large dataset).

7.2 Prediction by Time of the Day

The second improvement we did is to predict travel time using data from the same time of the day, since travel time for a route happening at 5AM will be very different from the same route happening at 5PM. We conducted the optimization for the first model. We tried two ways to divide the input data: into 24 hours and into parts of the day (before dawn 0-5, morning 6-12, afternoon 13-19, evening 20-24).

For the first model, predictions using hourly data and part-of-the-day data achieve better results than predictions by day. At hour 23, there is only 7 data points, so the error is extremely small. Part-of-the-day predictions are more stable than hourly predictions. Hourly predictions do badly at 2, 8, 16 o'clock, while part-of-the-day predictions do badly during morning hours. Results is shown in Figure 9.

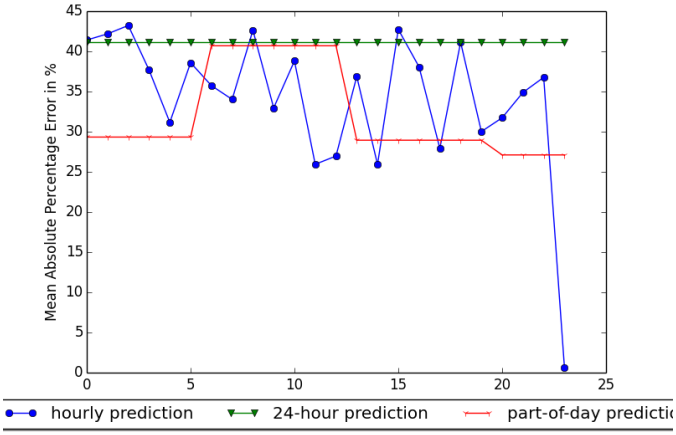


Figure 9: MAPE comparison for the first model for predictions using different time of the day.

7.3 Customize Loss Function

The loss function we used for all models is Mean Squared Error loss. This loss function penalizes positive and negative errors equally. We can relax this restriction. In real life, if model predicts arrival time earlier than the actual bus arrival time, it is acceptable because one or two minutes in advance will urge bus riders to arrive at the stop early and catch the bus on time. However, if predicted time is advanced too much, it will actually drive riders away. On the other hand, it is unacceptable to make predictions after the actual bus arrival time. Considering this, we can modify our loss function to be more realistic:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[(\hat{y}_i - y_i)_+^2 + \alpha * (y_i - \hat{y}_i)_+^2\right]$$

While training, α is set at 0.5. We tested the customized loss function for all three models and get following results in table 11.

Model	mean of y (m)	std of y	MBE	MaxAE	MeanAE	MAPE	RMSE	RSSE
M1.1	54.6680	23.6543	-12.8299	96.1293	22.3616	41.1286%	29.0639	296.5158
M1.2	42.117	23.9314	-5.9098	115.0933	11.8148	26.2901%	21.6167	208.1771
M1.3	43.1245	25.1217	-12.7370	132.3283	15.4744	28.9221%	27.2398	262.7963
M2-North	21.2333	12.6785	1.6543	1.9844	1.6543	10.5673%	1.6821	2.9135
M2-South	12.5285	6.8927	-4.4420	18.1242	5.9992	62.8067%	8.9979	21.9537
M2- East	5.4	0.7399	-1.3462	2.7640	1.6144	30.6459%	1.7821	3.9849
M2- West	5.7375	0.8170	-0.1590	1.8042	0.7731	13.3006	1.0037	2.2871
M2 - All	9.8957	8.8766	-2.4493	31.3964	3.8553	62.3219%	8.17095	36.1881
M3 - segment	9.1977	9.7143	-0.9721	51.8333	2.0190	24.0846	5.2978	93.6795
M3 - route	4.9265	20.1482	-2.9401	55.0181	4.9265	17.7262	10.063	102.1323

Table 11: Metrics for models using customized loss function.

8 Discussion and Conclusion

Starting with very intuitive model that predicts the route travel time using bus GPS data, we obtained a not very good result. With more prior knowledge added to the model, the predictions accuracy improves. Making the route travel time prediction model use the segment travel time prediction model achieves better performance, since the preceding buses implicitly embed the traffic information. It also outperforms segment travel time prediction model, because negative and positive errors of segment predictions cancel out. Moreover, grouping route traversal data by parts of the day and hour of the day and making predictions only using data from the same time slot of the day improve model accuracy. It tells us that for time-sensitive data like bus travel time, morning route traversal situation is very different from evening ones, so we should never mix them together to make predictions.

In addition, utilizing a customized loss function makes the predictions better fit bus riders' needs. And it is promising to see that with more data and including more factors in the input features, the model is able to generate better results, thus making the model of practical value.

9 Acknowledgments

Thanks Professor Manuela Veloso for guidance on the directions of this project, model ideas, improvements, and advice on writing this report. Thanks Professor Susana Sargento for sharing this dataset. Thanks Jorge Pereira for detailed documentation to make this dataset clear.

10 Appendix

10.1 Feature Description

Feature Name	Description
node_id	ID of the installed boards in the vehicles
system_time	Time instance of the board, generally the same as GPS time.
gps_time	Time instance when the information was captured
server_time	Time instance when the inforamtion reach the server
latitude	Latitude of vehicle
longitude	Longitude of vehicle
altitude	Altitude of vehicle
speed	Speed of vehicle in km/h
heading	Heading/direction of the vehicle (Degrees)
hdop	Horizontal Dilution of Precision
accel_x	Acceleration of x axis
accel_y	Acceleration of y axis
accel_z	Acceleration of z axis
hops	Number of hops until information reaches server (minus 1 when information sends directly through as RSU)
network	Connection network (802.11p, cellular)
next_hop_id	next node id (OBU or RSU) where the traffic passed after being sent
rsu_id	ID of the RSU that received the information
rssi	Received Signal Strength
vehicle_stats	Vehicle status (1-Engine started; 2-Moving; 3-Stopped; 4-Dead; 5-Unknown)
distance	
connected_time_p	Time connected through WAVE network (.11p) in seconds
connected_time_3g	Time connected through Cellular network in seconds
traffic_3g_in	Incoming traffic made through Cellular (4G) network
traffic_3g_out	Outgoing traffic made through Cellular (4G) network
traffic_g_in	No entries
traffic_g_out	No entroes
traffic_p_in	Incoming traffic made through 802.11p network
traffic_p_out	Outgoing traffic made through 802.11p network
traffic_eth_in	Incoming traffic made through physical netowrk, could be active in RSUs
traffic_eth_out	Outgoing traffic made through physical network, could be active in RSUs
traffic_3g_I2_in	Cellular traffic layer 2
traffic_3g_I2_out	Cellular traffic layer 2 outgoing

10.2 Sample Data Point

Feature	Example 1	Example 2	Example 3
node_id	195	215	372
system_time	2015-04-05 00:00:00	2015-04-05 00:00:00	2015-04-05 00:00:00
gps_time	2015-04-05 00:00:00	2015-04-05 00:00:00	2015-04-05 00:00:00
server_time	2015-04-05 01:01:00	2015-04-05 01:02:00	2015-04-05 01:01:00
latitude	41.148689	41.173859	41.183071
longitude	-8.610642	-8.619491	-8.621082
altitude	100	109	99
speed	0	66	0
heading	190	265	222
hdop	0	0	0
accel_x	0	0	0
accel_y	0	0	0
accel_z	0	0	0
hops	1	0	2
network	p	3g	p
next_hop_id	481	0	501
rsu_id	481	0	447
rssi	42	0	21
vehicle_stats	1	5	1
distance	0	294	0
connected_time_p	13	2	13
connected_time_3g	0	13	0
traffic_3g_in	0	0.076	0
traffic_3g_out	0	0.076	0
traffic_g_in	0	0	0
traffic_g_out	0	0	0
traffic_p_in	4842.691	0	1.439
traffic_p_out	259.822	0	2.041
traffic_eth_in	0	0	0
traffic_eth_out	0	0	0
traffic_3g_I2_in	0	0.076	0
traffic_3g_I2_out	0	0.274	0

References

Gurmu, Z. K. and Fan, W. D. (2014). Dynamic travel time prediction models for buses using only gps data. In *Transportation Research Board 93rd Annual Meeting*, number 14-0378.

Hall, R., Dessouky, M., Nowroozi, A., and Singh, A. (1997). Evaluation of its technology for bus timed transfers. *California Partners for Advanced Transit and Highways (PATH)*.

Lin, W.-H. and Zeng, J. (1999). Experimental study of real-time bus arrival time prediction with gps data. *Transportation Research Record: Journal of the Transportation Research Board*, (1666):101–109.

Wang, R., Veloso, M., and Seshan, S. (2014). O-snap: Optimal snapping of odometry trajectories for route identification. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5824–5829. IEEE.

Yu, B., Lam, W. H., and Tam, M. L. (2011). Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies*, 19(6):1157–1170.