Marius Latinis

Bus Arrival Time Prediction

Computer Science Tripos – Part II

Christ's College

May 1, 2017

Proforma

Name:	**Marius Latinis**
College:	**Christ's College**
Project Title:	**Bus Arrival Time Prediction**
Examination:	**Computer Science Tripos – Part II, July 2017**
Word Count:	**TODO(ml693): figure out**
Project Originator:	**Dr Richard Mortier**
Supervisor:	**Dr Richard Mortier**

Original Aims of the Project

- Implement the prediction algorithm. The GPS points showing how the bus has moved so far are given as an input. The output of the prediction algorithm is the predicted time when this bus will arrive at the future stop.
- Evaluate the prediction algorithm and present the results.

Work Completed

The following milestones have been achieved:

- The prediction algorithm is implemented.
- The algorithm is evaluated and the results are presented in the evaluation section.
- (Optional) the real-time prediction system is running.

Special Difficulties

None

Declaration

I, Marius Latinis of Christ's College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed [signature]

Date May 1, 2017

Contents

Chapter 1

Introduction

1.1 Motivation

In the European countries, public buses are a popular form of transportation. People want to arrive in time to a particular place and they use the public transport for that. For example, a bus is used to travel to the airport, arrive at the first lesson in school. Citizens rely on a vehicle to travel a certain distance in a certain amount of time. The problem arises when the bus fails to arrive in time due to the traffic congestion. This failure makes passengers miss various things. For instance, it is very dissapointing to miss a departing train due to a bus being late. Therefore, people are looking for a source of information that can reliably tell when a vehicle will reach a certain place.

1.2 Timetable

The timetable is one potential source of information:

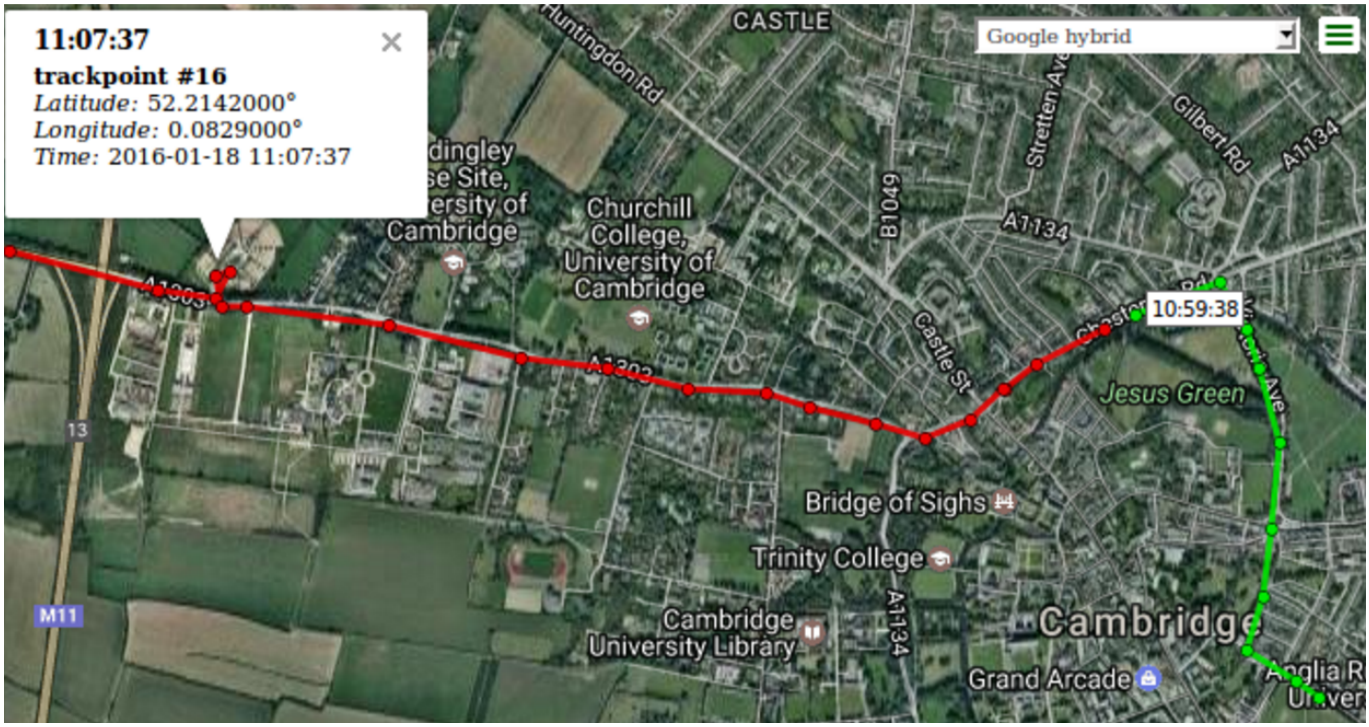
Often the timetable appears as a stand near the bus stop S_i. It can show multiple things, such as the bus arrival time to the same stop S_i, or a bus arrival time to the next stops $S_{i+1}, S_{i+2}, ...$ The way to present a timetable also varies. For example, it is very likely that the information present on a stand is also displayed in some website.

Unfortunatelly, every timetable is fundamentally limited. The arrival times it presents are **static**. If the timetable claims that the bus will arrive at the stop at time T, the value T will not change. Such static arrival time prediction fails to reflect the dynamic aspect of the transport.

1.3 GPS

The current technology allows making use of the GPS data. Modern buses are equipped with the devices sensing their GPS position. The device sends the position at regular intervals to a determined server that can store and instantly analyze the data. In particular, the prediction analysis can be done to guess when the bus will arrive at the next stop. The goal of my project is to perform such prediction analysis. I aim to build an algorithm which predicts the bus arrival times based on the most recent GPS data.

1.4 Actual Problem Overview



The most recent GPS points (indicated in green in the diagram above) show where the bus has travelled up to now. The future data (indicated in red) is not known during the prediction phase. My goal is to predict the future data given the present data.

In this specific example I want to predict when the bus will reach the Madingley Park starting from the Chesterton Road. One can see it takes about 8 min. (10:59–11:07) to travel such distance.

Chapter 2

Preparation

2.1 Existing Prediction Systems

Traffic varies a lot based on the vehicle type, the city size, the road infrastructure present in the city and other factors. There is no one universally accepted algorithm that predicts well for each of the different cases. For example, the train arrival time can be predicted dividing the remaining travel distance by its current speed. This works well for trains. No other vehicle is blocking the train driving on the rail and its speed is constant for most of the trip. However, such approach is hopeless for buses in the busy city. The current speed will quickly change due to the upcomming turns and traffic lights. Hence, just the momentary speed alone is of little use.

Thus, it is important to code not any algorithm, but the one which suits the GPS data and the type of traffic I am dealing with. I did a thourough preparation to choose the correct approach.

Firstly, I read two papers describing how other scientists are solving this problem. One of them is:

- Bin Yu, William H.K. Lam, Mei Lam Tam: Bus arrival time prediction at bus stop with multiple routes

  http://www.sciencedirect.com/science/article/pii/S0968090X11000155

On a high level, the paper uses the most recent buses that arrived to the bus stop B from some location A. It feeds the times $t^i_{A->B}$ (how long it took for the i-th preceding bus to reach B from A) into the machine learning model (neural network, support vector machine and others are tried) and hopes that the model will predict the sensible arrival time for the current bus to reach B from A.

My implementation incorporates several ideas from that paper:

"[...] the running time(s) of the preceding bus(es) that has(ve) **just** reached the stop can be used to reflect the traffic conditions. Furthermore, [...] the weighted average running times of several preceding buses could reduce the effect of accidents on the preceding buses."

However, after discussing with Prof Kelly, I decided not to bother doing Machine Learning and continued searching for an alternative. The problem many Machine Learning models have is that they act like a black box. For example, the weights on the edges of a neural network eventually converge to certain values. It is often difficult to understand the intuitive meaning of a particular value (e.g. why this weight equals 1.2 but not 1.4?). It is thus hard to explicitly re-program the model in case it does not work on the data.

1. Pengfei Zhou, Yuanqing Zheng, Mo Li: How Long to Wait? Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing

   http://www.ntu.edu.sg/home/limo/papers/sys012fp.pdf

I learned the common evaluation metrics used by reading these papers. I also looked at the results claimed in those papers to get a first clue how well the current prediction systems are working.

Secondly, I talked to multiple people in Cambridge. Several lecturers gave me good tips. For example, my overseers pointed out that knowing the exact streets a bus follows is not neccesary for making a prediction. Only the GPS points where the bus has moved so far and the geographic location of the future bus stop can be enough to predict the arrival time to that stop. This advice saved a lot of my time as I did not have to search for a good map data.

Finally, after talking with Prof Kelly I became convinced that building a simple algorithm described in this dissertation is the right way to start solving the arrival time prediction problem.[1]

2.2 Starting Point

Initial Starting Data

A company named Vix is sending the real-time GPS data to the Cambridge servers. The GPS data shows how certain[2] East-England buses are moving in the real-time. The Cambridge servers convert the binary GPS data into a human readable JSON format. The JSON data is saved. At the beginning of the academic year I was given the JSON data spanning 3 months (June, July and August 2016). That was the starting point data.

To start with, I wrote the prediction algorithm entirely from scratch and tested it on the historical starting point data.

Starting Data for the Real-time System

In the second term, I started building the real-time system as an optional extension. The optional part requires the real-time GPS data. To start with, Dr. Lewis gave me the access to the machine receiving the new GPS data from the Vix company every 30sec.

The code receiving the real-time GPS data and immediatelly converting it to the JSON format was already written. I started this part by writing the new code processing the real-time JSON data.

[1] Other lecturers were advising me to use the more sophisticated techniques, such as Machine Learning or Multivariate Analysis. I decided to discard these techniques in favour of a simpler solution.

[2] Stagecoach and Whippet buses.

Chapter 3

Implementation

3.1 Data Preprocessing

Below is the starting point input data example:


```
"entities":[{"received_timestamp":1476477040,"vehicle_id":"122","label":"ATS-3603","latitude
":51.91048,"longitude":-0.5006834,"bearing":150.0,"timestamp":1470092134},{"received_timesta
mp":1476477040,"vehicle_id":"132","latitude":51.891663,"longitude":-0.45255125,"bearing":6.0
,"timestamp":1470091865},{"received_timestamp":1476477040,"vehicle_id":"187","latitude":51.4
93637,"longitude":-0.14694783,"bearing":282.0,"timestamp":1470092143},{"received_timestamp":
1476477040,"vehicle_id":"193","latitude":51.909676,"longitude":-0.5108846,"bearing":18.0,"ti
mestamp":1470092134},{"received_timestamp":1476477040,"vehicle_id":"194","latitude":51.88216
8,"longitude":-0.41519493,"bearing":114.0,"timestamp":1470092134},{"received_timestamp":1476
```


One file contains a single snapshot of all buses, with one entry per bus. An entry such as:


```
vehicle_id":"132","latitude":51.891663,"longitude":-0.45255125,"bearing":6.0,"timestamp":1470091865}
```


means that a vehicle nr. 132 was at the geographical position $(51.891663, -0.45255125)$ on the 2nd August 2016, GMT time 01:51:05[1].

A new snapshot file arrives once in 30s. The data preprocessing part takes multiple input files[2] and converts them into another format:

File day15/bus300/trip2

```
time,latitude,longitude
2016-10-15 10:37:01,52.18485,0.20276
2016-10-15 10:37:32,52.18530,0.19788
2016-10-15 10:38:01,52.18548,0.19515
2016-10-15 10:38:32,52.18530,0.19343
2016-10-15 10:39:02,52.18557,0.19170
2016-10-15 10:39:32,52.18432,0.19070
2016-10-15 10:40:02,52.18271,0.19012
2016-10-15 10:40:32,52.18253,0.18840
2016-10-15 10:41:01,52.18297,0.18423
2016-10-15 10:41:31,52.18584,0.18509
2016-10-15 10:42:30,52.18647,0.18653
```

File day15/bus400/trip3

```
time,latitude,longitude
2016-10-15 12:01:35,52.20215,0.12460
2016-10-15 12:01:35,52.20215,0.12460
2016-10-15 12:02:35,52.19937,0.12705
2016-10-15 12:02:35,52.19937,0.12705
2016-10-15 12:03:35,52.19543,0.13064
2016-10-15 12:03:35,52.19543,0.13064
2016-10-15 12:04:34,52.19462,0.13423
2016-10-15 12:05:34,52.19444,0.13581
2016-10-15 12:06:35,52.19274,0.13581
2016-10-15 12:10:35,52.19247,0.13567
2016-10-15 12:11:35,52.19256,0.13409
```

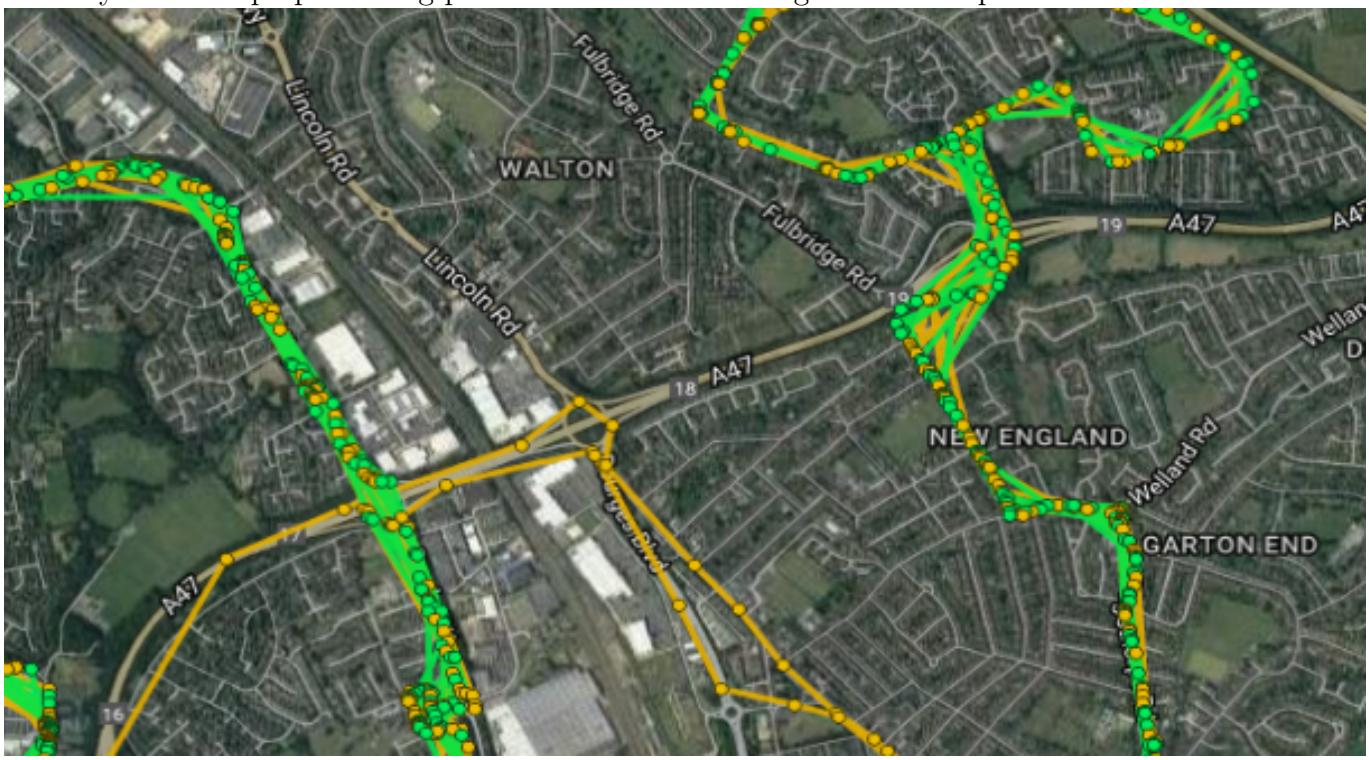
At the end I have a file for each trip made by one bus in one day[3].

[1] The time is calculated from the timestamp field.

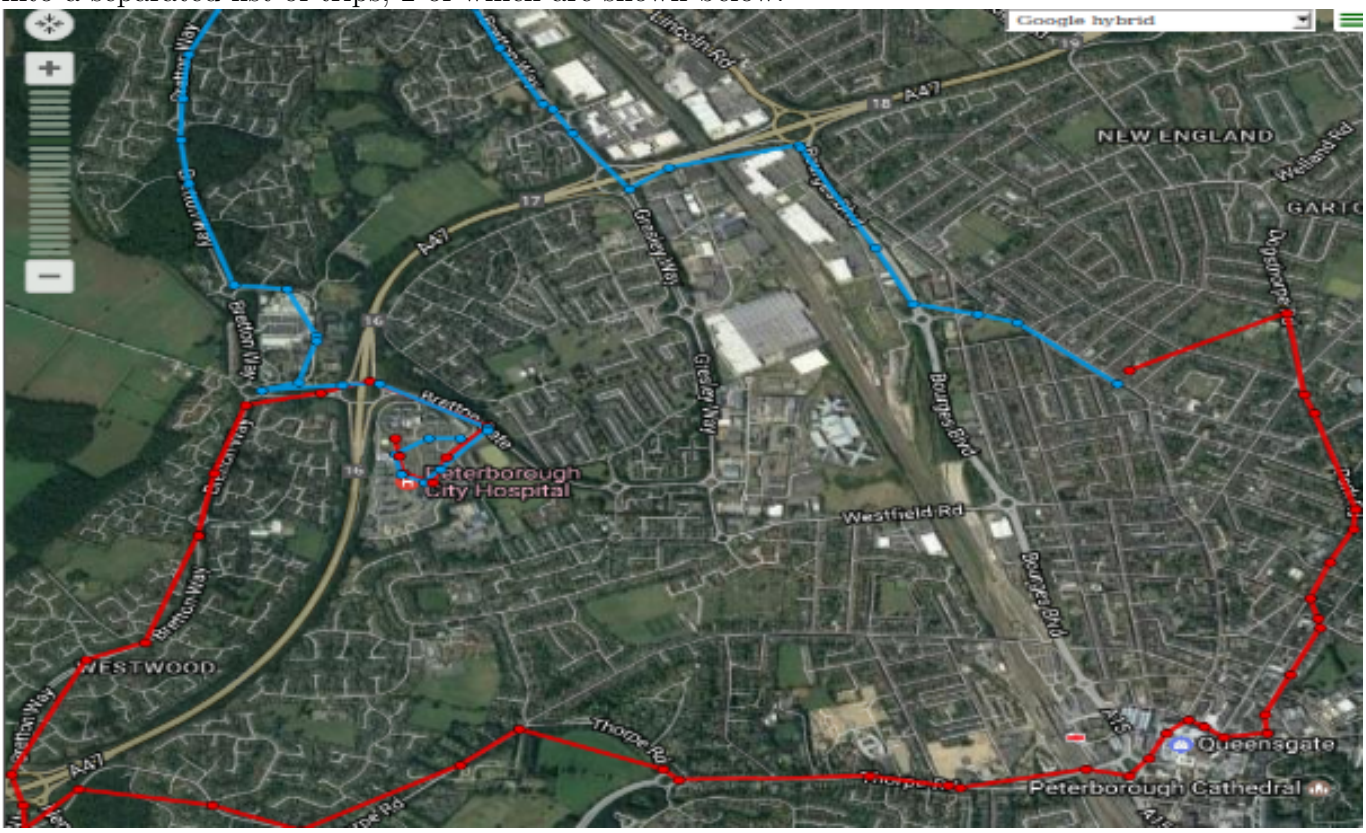
[2] For example, to build historical trips, I take $3600 \cdot 24/30 = 2880$ files received in **one** day and preprocess them.

[3] Note that a bus on a particurlar day could have made multiple trips.

Visually the data preprocessing part converts the following set of GPS points



into a separated list of trips, 2 of which are shown below:

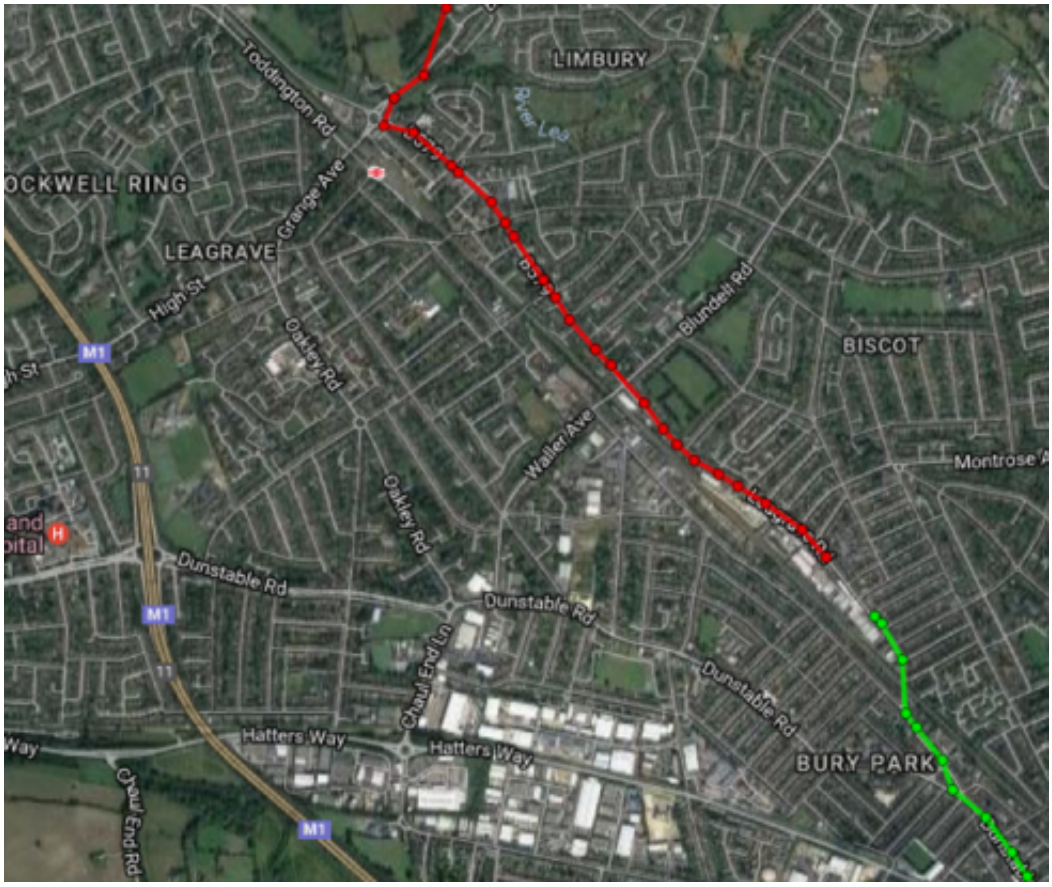


An optional use[4] of the preprocessing part is the help it provides to visually inspect the data. I am able to look at each trip separately (or to merge multiple trips of my choice) and understand why the buses are moving in such a way.

[4]The main use is that the preprocessing part is one of the components of the whole arrival time prediction system.

3.2 Route Detection Algorithm

3.2.1 Why an Algorithm is Needed



The GPS points indicated in green show how the bus has moved so far. The red points show where the bus will travel in the future, a path I do not know yet. Predicting when the bus will arrive at the stop requires knowing **where** a bus will travel next. How?

Approach Using Static Data

A standard way to find out where the bus will travel next is to know its route in advance. Some GPS entries contain additional fields, such as the **label** field:


```
"vehicle_id":"122","label":"ATS-3603","latitude":51.90439,"longitude":-0.5070053,
```


An extra field **label** names the route a bus follows. Auxiliary data might indicate what route the label **ATS-3603** corresponds to. One can then look at the route and determine a sequence of stops a bus will visit next.

Unfortunately, this approach has drawbacks:

- Not all entries contain auxiliary fields. After all, the GPS device on the actual vehicle is responsible only for providing the latitude and longitude coordinates, not the route a bus follows.
- Good static data to infer routes based on additional fields is not straightforward to find. Field values might not match (For example, a route named **SW-4** in the real-time data is actually

SWX-4). Buses are often changing routes[5] and data does not always indicate that.

- The static data only shows the sequence of bus stops, not a detailed sequence of streets a bus will follow. It is much better to know an actual path for the accurate arrival time prediction.

New Approach Overview

To overcome the static data limitations, I propose a new way to infer the route a bus follows. I will align a sub-trip how the bus has moved so far with a historical path for which the route is known. The path aligning best to the current sub-trip will be used to predict where a bus travels next. To perform the alignment, I constructed an algorithm[6] that can tell whether one sequence of GPS points follows another.

3.2.2 *FollowsPath* Predicate Definition

I define a predicate $FollowsPath(sub\text{-}trip, path)$ to be true iff a *sub-trip* follows a *path*:

Two pictures show when I want the predicate to return true and when false. For the first example, the predicate returns true because it is intuitively clear that a red sub-trip follows a blue path. In the second example, the output is false, as the two GPS sequences do not intersect at all.

[5] A bus is unlikely to follow the same route every day

[6] From now on referred to as a boolean predicate.

3.2.3 *FollowsPath* Predicate Implementation

Firstly, consider the case when a sub- trip's GPS points (P_0, P_1, ..., P_{n-1}) **exactly** follow another path:

Each point P_i is on some segment of another trip's path. For example, P_1 is on the segment AB. Let $x = |AP_1|$ and $y = |P_1B|$. Then the error[7] function defined as

$$err(P_1, AB) = \frac{x+y}{|AB|} - 1$$

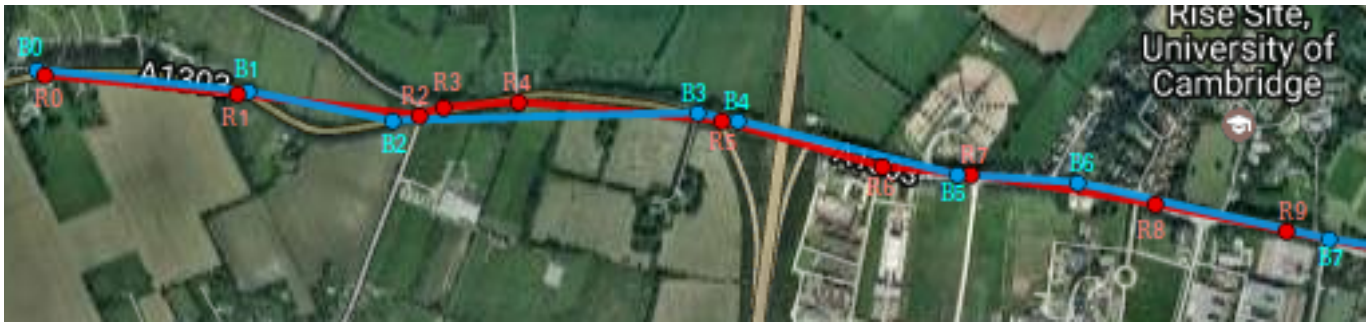
equals 0. For each point P_i, I find a segment S_{P_i} such that $err(P_i, S_{P_i})$ is minimised. The *FollowsPath* predicate returns true iff

$$S = \sum_{i=0}^{n-1} err(P_i, S_{P_i}) < n\epsilon$$

for a suitably chosen threshold value ϵ ($\epsilon = 0.1$ is one choice).

Note that $S = 0$ when a sub-trip follows a path exactly. Therefore, in this case, the *FollowsPath* predicate returns true, which is what I want.

The case when a sub-trip does follow a path, but not exactly (the real world situation), is illustrated in the following picture:



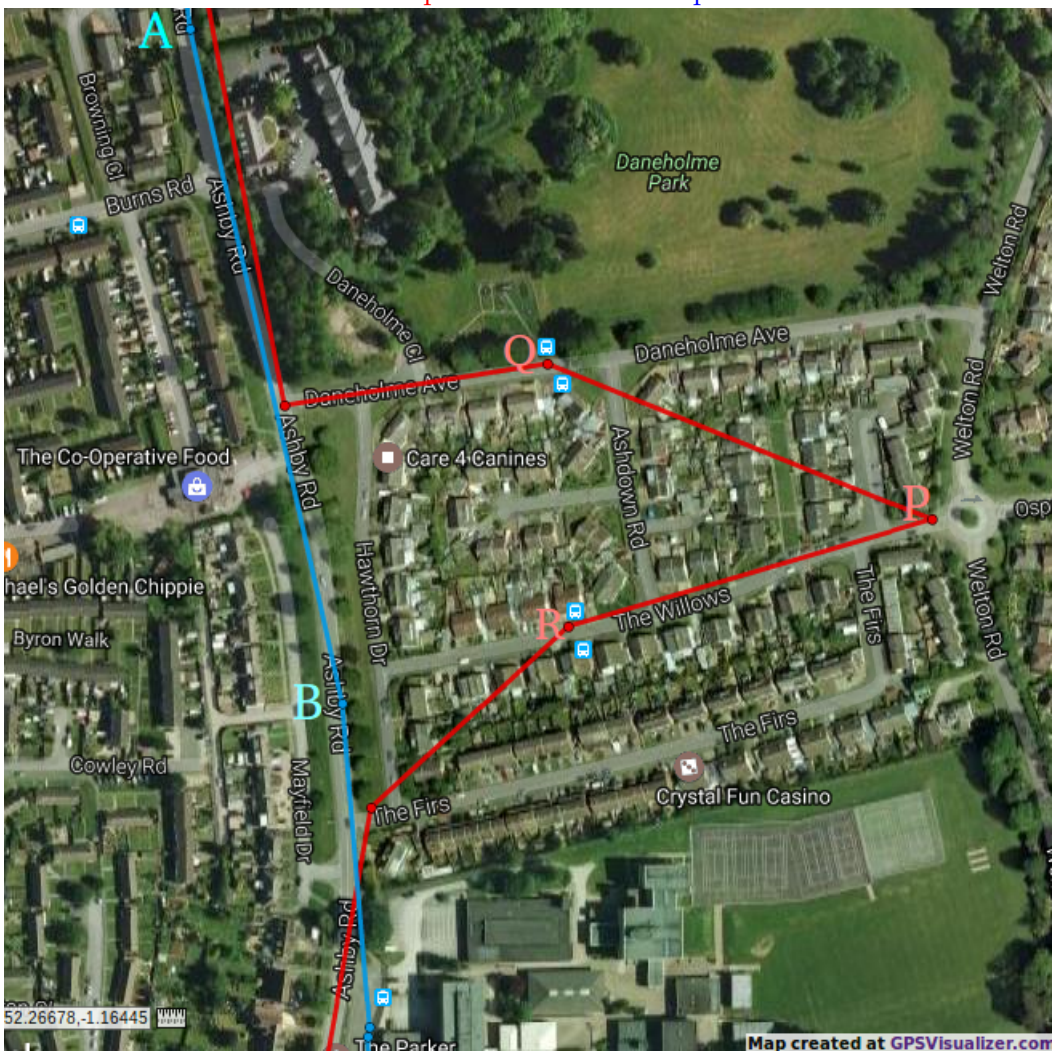
In this example I am aligning a sub-trip (R_0, ..., R_9) with a path (B_0, ..., B_7, ...). The computation of S proceeds as follows:

$$\begin{aligned} S &= err(R_0, B_0B_1) + err(R_1, B_0B_1) + err(R_2, B_2B_3) + err(R_3, B_2B_3) + err(R_4, B_2B_3) + \\ &\quad err(R_5, B_3B_4) + err(R_6, B_4B_5) + err(R_7, B_5B_6) + err(R_8, B_6B_7) + err(R_9, B_6B_7) \\ &= (1.0294 - 1) + (1.0300 - 1) + (1.0021 - 1) + (1.0730 - 1) + (1.0208 - 1) + \\ &\quad (1.0714 - 1) + (1.0286 - 1) + (1.0158 - 1) + (1.0126 - 1) + (1.0125 - 1) < 10 \cdot 0.1 = n\epsilon \end{aligned}$$

and the predicate again returns true, as desired.

[7] error function takes a point and a segment as an input

The last case is when a sub-trip does not follow a path:



In this case the triangle inequality ensures large $err(P_i, S_{P_i})$ values for certain points P_i. For example, for the point P in the diagram, the smallest error value is

$$err(P, AB) = (12.9 + 9)/10^8 - 1 = 1.1$$

This error is more than 10 times larger than the typical values being less than 0.1. Furthermore, large errors will be generated for each sub-trip's point that deviated from the path, such as for the points Q and R in the diagram. Hence, the total errors sum S will exceed $n\epsilon$ and the predicate will return false.

The practical evidence that the $FollowsPath$ predicate works is given in the evaluation section.

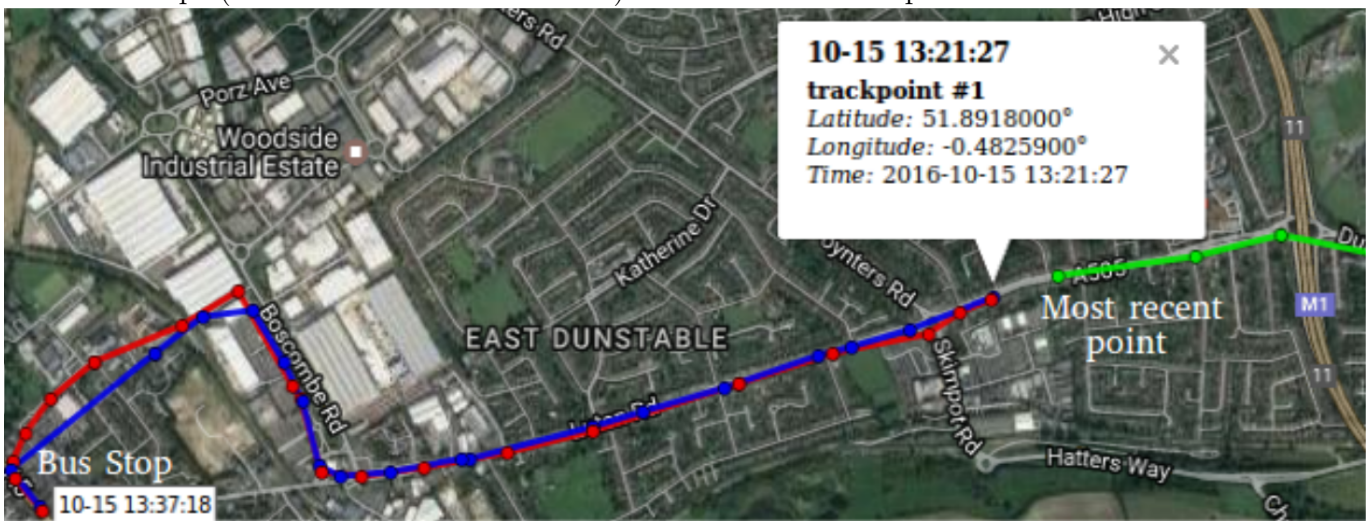
[8]The distances 12.9, 9 and 10 are scaled here. Scaling does not change the ratio value.

3.3 Arrival Time Prediction

3.3.1 Basic Idea

The *FollowsPath* predicate allows to detect the route a bus follows. The route tells the next stops a bus will visit. This section shows how to predict the arrival time to each future bus stop.

Historical trips (which follow the same route) are used to make a prediction:



In the picture above 2 example historical sub-trips (blue and red) are shown. Both of them start where the vehicle is currently located (indicated using the most recent green point). Both of them represent the history how past vehicles travelled until the bus stop. I compute the duration how long it took for each historical trip to reach the bus stop. Knowing multiple duration values I return the median as the predicted arrival time.

For example, it took 13:37:18 − 13:21:27 = 15m 51s for the blue trip to reach the bus stop. I also calculate these duration values for the red and other trips present in the historical trips set. The computation gives me the list of numbers. I sort the list:

(8m 30s, 14m 46s, 14m 58s, 15m 06s, **15m 36s**, 15m 51s, 16m 07s, 16m 31s, 17m 34s)

and return the median value. Thus, I predict that the bus indicated in green will reach the bus stop after 15m 36s.

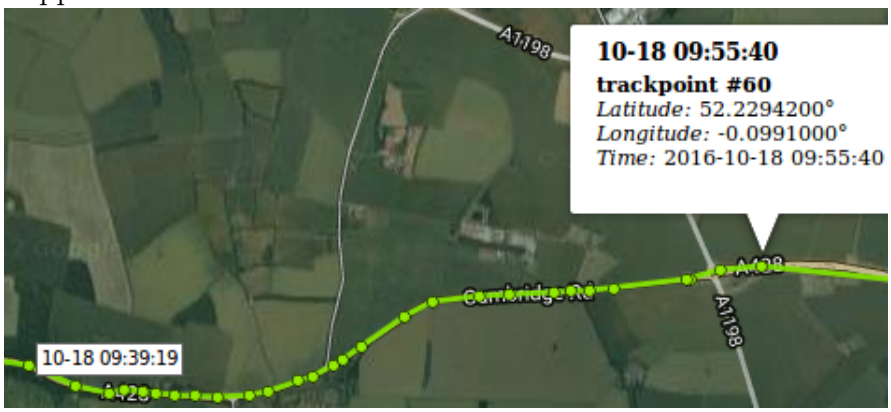
The reason for choosing a median value rather than some other average number (such as the arithmetic mean) is to get rid of the outliers. The shortest arrival time 8m 30s is a much smaller number than others. I discard this number when computing the median. Different computation might have been negatively influenced by the outlier. The median value predicted better compared to the arithmetic mean during the testing phase. Thus, I decided to stick with the median value.

3.3.2 Optimisation Nr. 1

If in the historical trips set certain trips have **just** travelled along the same route (e.g. in the last 30min.), I predict the median time just from those recent trips. Otherwise, the basic approach is used.

3.3.3 Optimisation Nr. 2

Suppose the most recent data



shows the distance a bus has travelled in the last 16 minutes. Certain historical trips

have travelled the same distance in a very different amount of time[9] (e.g. 4 min.). Predicting the arrival time using these historical trips can be misleading. This optimisation discards misleading trips.

I pick the last N points (e.g. $N = 30$) of the trip I want to predict the arrival time. The 1st point out of N shows where the bus was M minutes ago (if $N = 30$, then most likely $M = 15$, because the new GPS point is usually generated once in 30s). For each historical trip I find the point P where the historical bus was M minutes ago from the moment of time when it reached the current bus location. If P is roughly the same location as the current bus location M minuts ago, I include the historical trip into the set C of consideration. I discard this historical trip otherwise. After the set C is generated, I proceed only with it as before (3.3.1).

This optimisation is the simplest way I found to take into account the traffic periodicity. One can expect that the traffic on Tuesday might be similar to the traffic on Monday (a period of one day). Thus, one can hope that historical trips of the last day are suitable to predict what will happen today. However, this is not always true. The traffic on Monday (the working day) is very different from the traffic on Sunday (the weekend). Setting a period to be 1 week also does not fix the problem entirely. It can be that the Monday a week ago was a holiday day. Traffic during holidays is also different from the traffic during the regular days.

[9]This could have happened due to the different traffic congestion level in a different time of the day.

Fortunatelly, my optimisation addresses both the fact that the traffic follows some periodic pattern and also the fact that the period is not strict. One way to look at the optimisation described above is as follows. The bus location M minutes ago shows its current **"speed" along the route** (not the actual mph velocity, but how quickly it passed the route segment). The speed is correlated with the traffic congestion. The reason why some historical trip (which happened either a day or a week ago) had similar speed to the current bus is because the traffic was similarly congested to how it is congested now. And if the trip was similarly congested because of the periodic pattern, then this optimisation detects all periodic trips, regardless what the actual period is.

Optimisations are tested in the evaluation section.

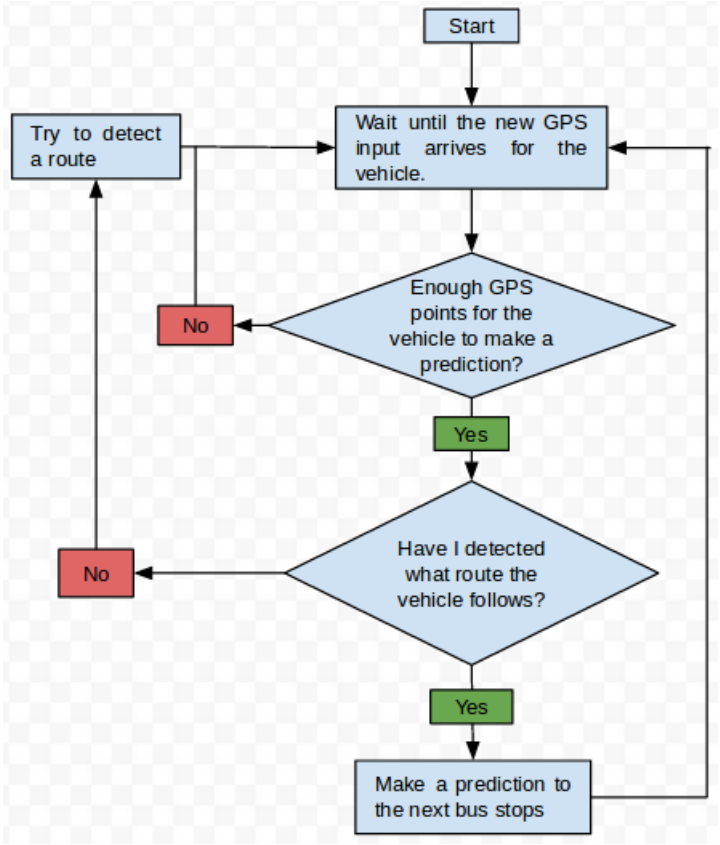
3.4 Real-time System (Optional)

I am processing the GPS data at real time. The goal of this chapter is to present the current system's state so that whoever comes after me[10] can understand the system and improve it.

One machine every 30s. accepts a new JSON file containing the GPS data. For each new file the machine does the following:

(a) Reads the new GPS data from the file and updates the arrival time prediction values.

Below is a flow chart handling **one** bus:



This is how for **each** bus I **independently** process its new GPS data and predict the arrival time to the next bus stops. Currently one machine is waiting for t sec., then using the new GPS data to handle 50 buses in $30 - t$ sec.[11]

(b) Saves the JSON file to a disc. Once in 24h reads the files saved on a disc and generates the new historical data to improve the future predictions.

The current state of a system is such that for each new day I save the JSON files to a separate directory. When a day $d + 1$ comes, I read all JSON files corresponding to day d and perform the data preprocessing part (3.1) on those files. This generates about 20000 new trips every day. For each of the trip generated I then detect which route it followed and save this trip into the historical trips directory corresponding to that route. Hence, after each day I generate new historical trips following a route. New historical trips improve the arrival time prediction performance (part 3.3).

[10] Such as another student next year.

[11] Of course, I am trying to minimise t and maximise the number of buses handled $30 - t$ sec.

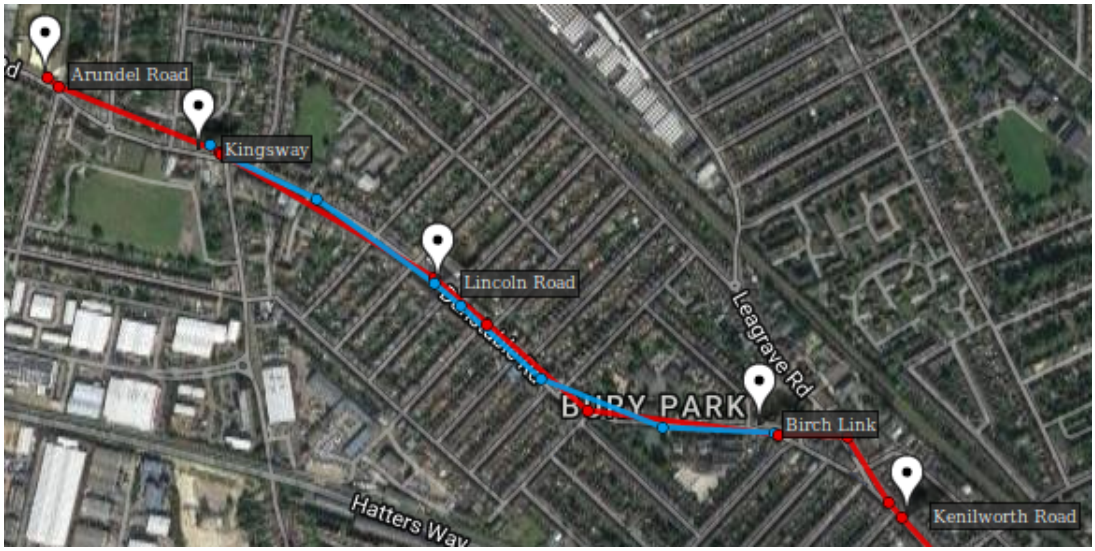
Chapter 4

Evaluation

4.1 Route Detection Evaluation

4.1.1 Sensitivity Score

The first step is to check whether the $FollowsPath(sub\text{-}trip, path)$ predicate returns *true* when the *sub-trip* indeed follows the *path*. I take the *path* 992088-20150526-20150830 to evaluate this step. I have a set T of 117 trips that went through it. For each trip $t \in T$ I take a *sub-trip* and check that the $FollowsPath(sub\text{-}trip, path)$ predicate returns true:

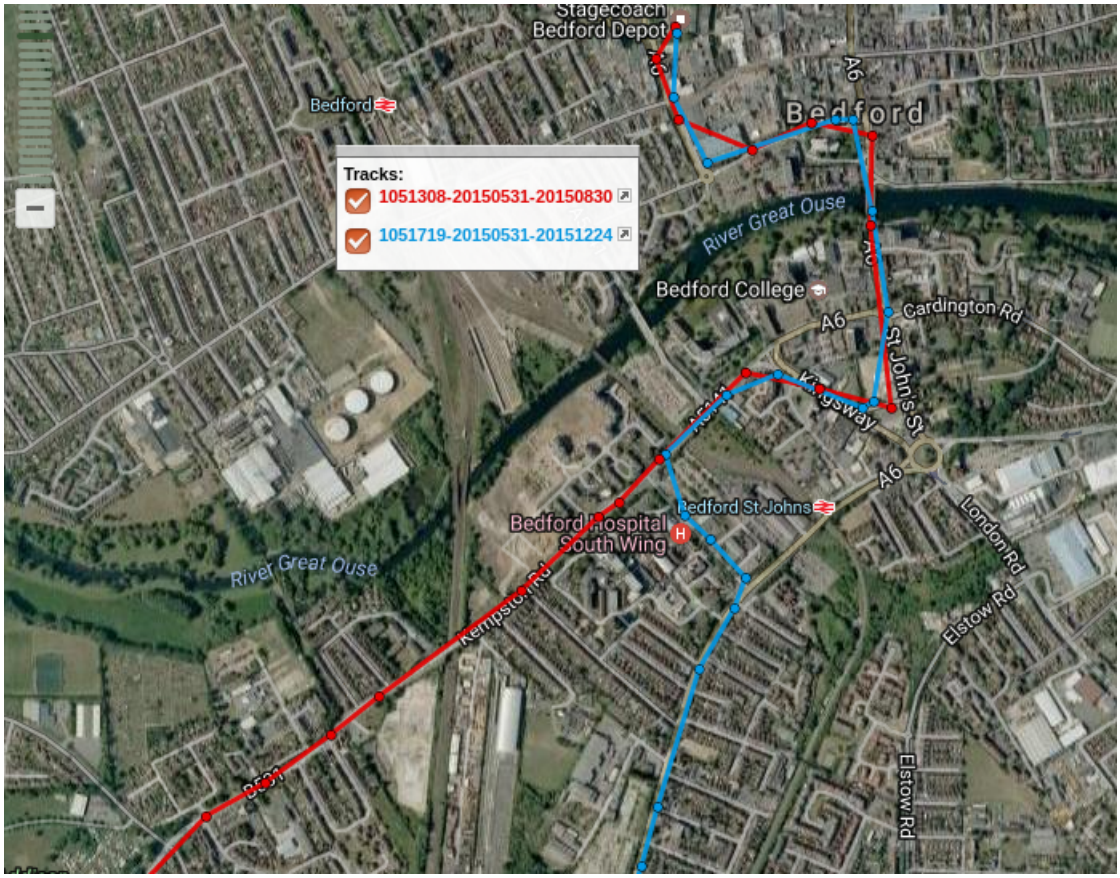


For this route the $FollowsPath$ predicate has 100% success rate.

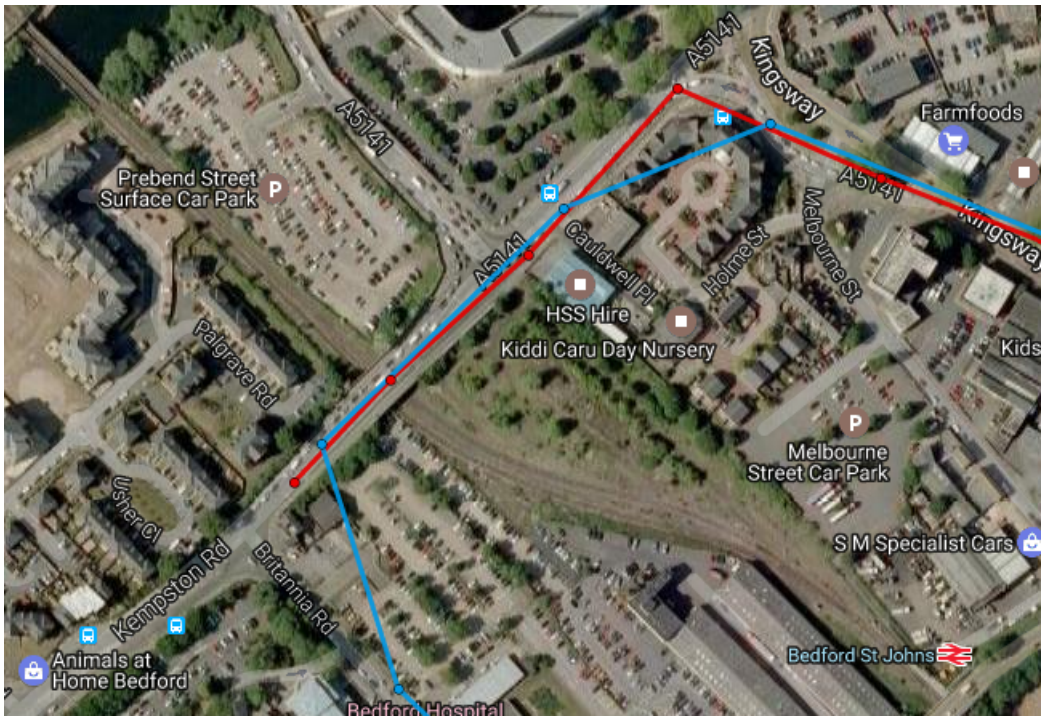
4.1.2 Specificity Score

The second step is to check that for trips not on a path the predicate returns false.

If a trip is very far away from the path (e.g. 10 miles away), the $FollowsPath(trip, path)$ will definitely return false. Thus, to perform a meaningful evaluation, I select a path 1051719-20150531-20151224 and a set of trips T following a very similar but nevertheless distinct path:



Both paths are the same at the beginning and separate later. For each trip $\in T$ I pick a *subtrip* of the first 12 points[1] and check whether the $FollowsPath(subtrip, path)$ returns false. Out of 107 trips only once the predicate incorrectly returns true. The misclassified *subtrip* has not deviated from a path yet for a predicate to detect that:



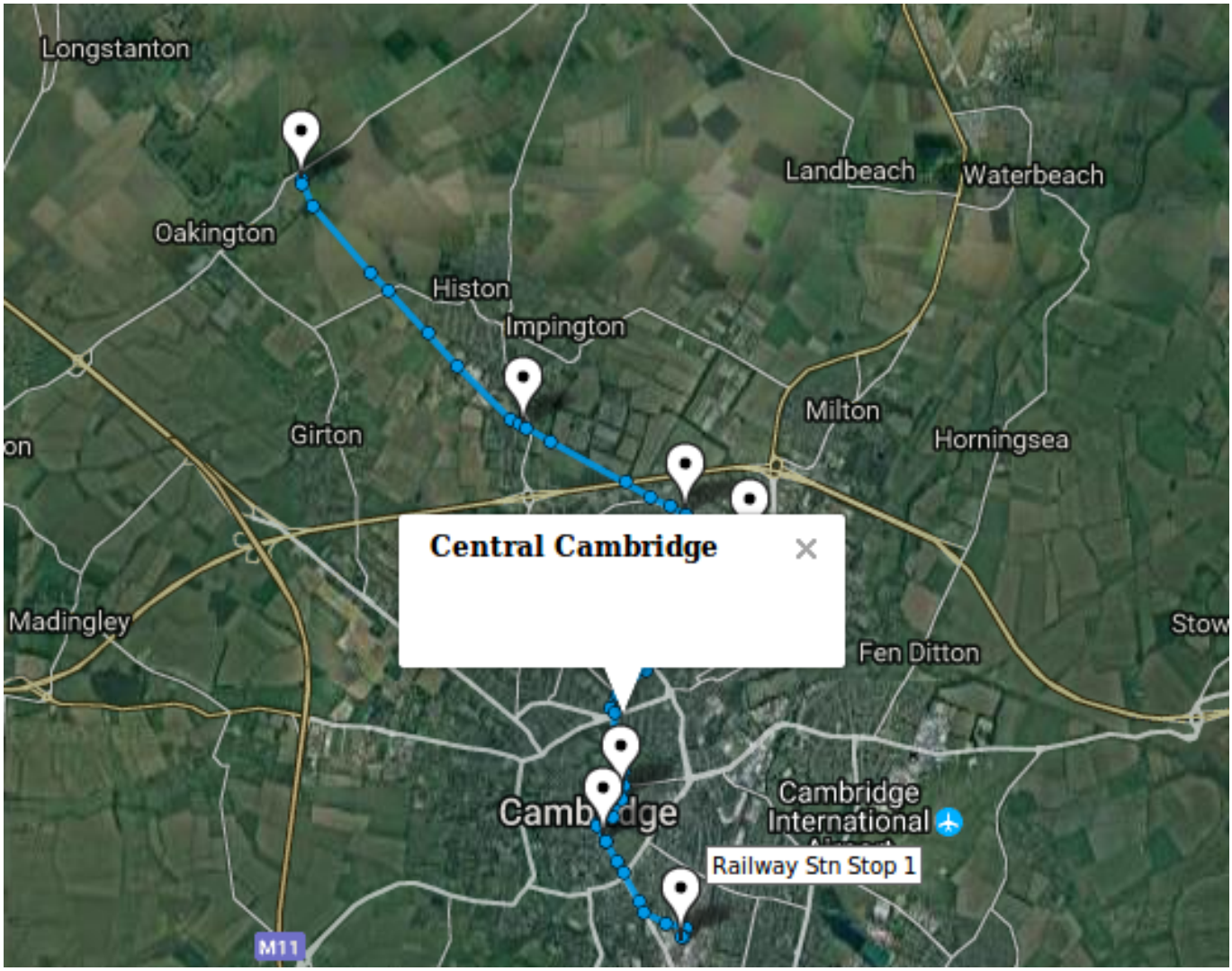
The sensitivity and specificity scores show that the $FollowsPath$ predicate works **if** used in the right place at the right time.

[1]Each trip deviates from a path approximatelly after 12 GPS points.

4.2 Arrival Time Prediction Evaluation

4.2.1 Evaluation Metrics Used

This evaluation section is the most important. It shows how well the algorithm predicts the arrival time. A few evaluation metrics are used repetitively throughout the section. Hence, I want to present them. For the presentation I will use the example route 1046531-20150531-20150830, which begins outside of Cambridge and ends at the Cambridge Railway station:



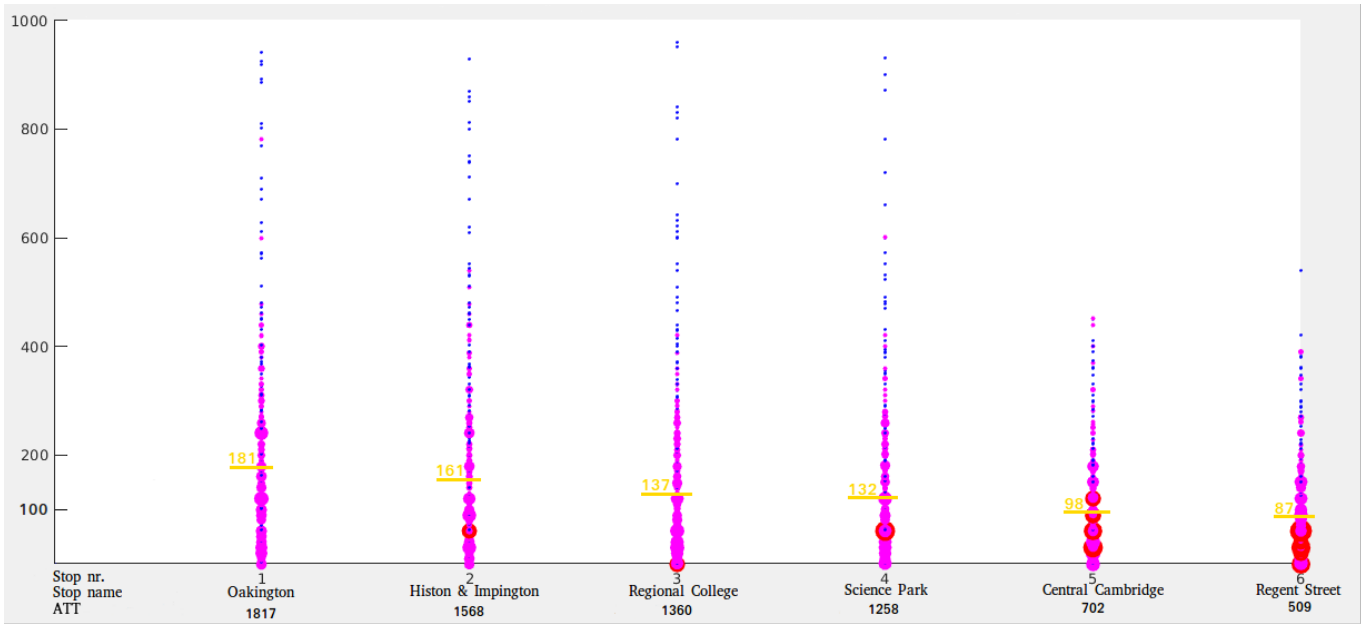
(i) **The MAE value**. Suppose I have a set of trips T, such that each trip eventually reaches a bus stop of interest. Suppose for trip $t \in T$ the algorithm predicts the bus arrival time at stop to be $t_{predicted}$, but the bus actually arrives at t_{actual}. The absolute prediction error for trip t is $E_t = |t_{actual} - t_{predicted}|$. The **mean** absolute prediction error for the whole set T is

$$MAE = \frac{\sum_{t \in T} E_t}{|T|}$$

For example, suppose I predict when the buses will reach the Railway Station starting from the Central Cambridge stop. I have 558 trips following this route to test my prediction. For each test trip, I assume only its history up to the Central Cambridge stop is known. I predict the arrival time to the Railway Station. After the prediction is made, I then read the rest of the history and check when the trip has actually arrived at the Railway Station. For this

example trip, the resulting MAE value is **98** sec. This number shows the average error the prediction algorithm makes. One can also use the MAE value to compare 2 algorithms. I claim that the prediction algorithm A performs better for some set than the algorithm B, if $MAE_A < MAE_B$.[2]

(ii) **The scatter plot**. The MAE value gives a first clue how well the algorithm performs. However, just the MAE value alone is misleading. 98 seconds can be either a small or a large error. It depends how far in the future I am predicting. It takes about 10 minutes to reach the Railway Station from the Central Cambridge stop. Thus, 98 seconds is an error **with respect to** the Central Cambridge stop. The scatter plot formalises this relation:

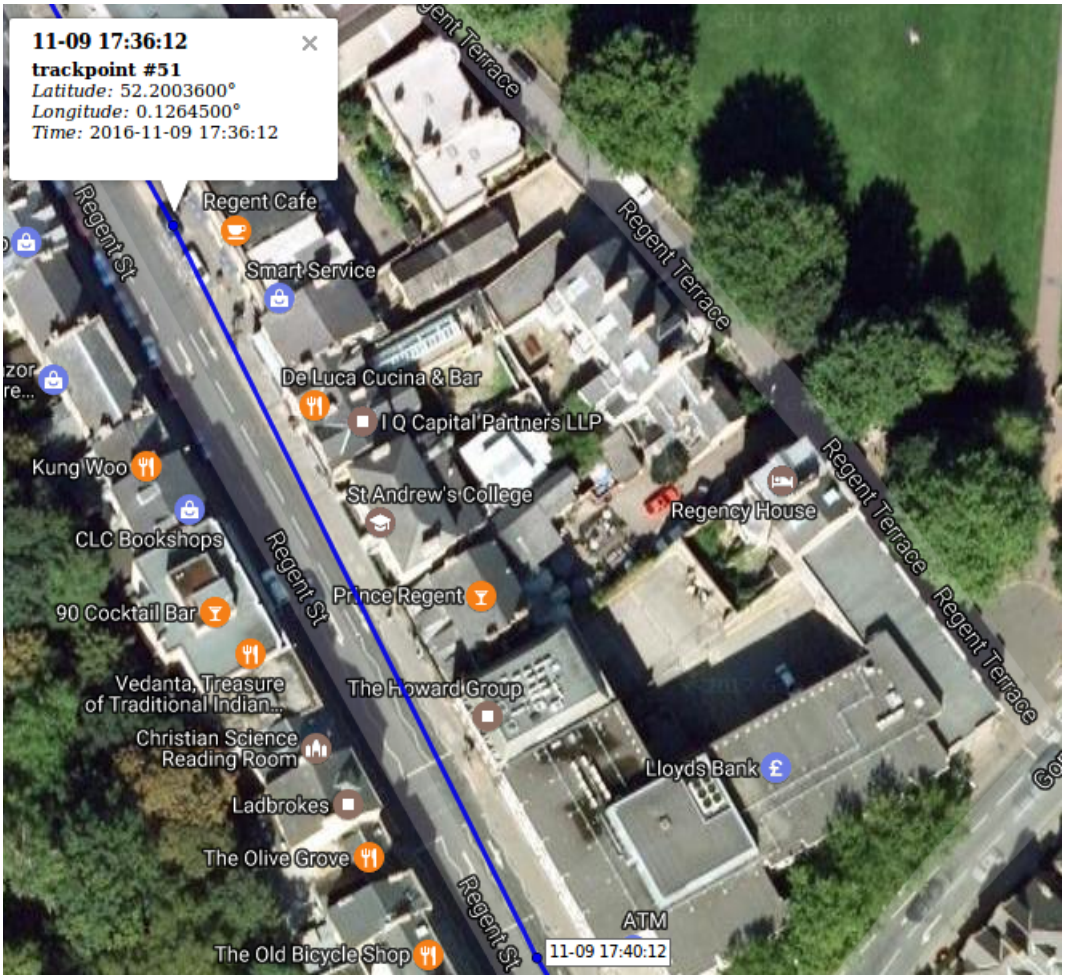


The x-axis labels the stop number i. For each i, I predict when the bus will reach the Railway Station (the last stop) starting from the i-th stop. The y-axis labels the absolute prediction error[3]. A point (i, y) tells that for some trip the prediction from the i-th stop to the last stop has an abolute error equal y. The more repeated errors (i, y) I have, the larger point I plot. For completeness, in the diagram I show the MAE value[4] and the average travel time[5] from the i-th stop to the last stop.

A few things can be observed in this plot:

- Firstly, the MAE value is smaller for each larger index i. The less time it takes for a bus to reach the last stop, the more accurate the prediction becomes.
- In my case, many prediction errors are smaller than the MAE value, except for a few large errors that make the MAE value increase. The large errors occur due to unusual traffic patterns:

[2]Such claim will be seen later.
[3]Measured in seconds.
[4]Labelled in yellow.
[5]For each trip I calculate the time it actually takes to reach the last stop, then return the mean value.



For some bus it took 4 minutes to pass just a few buildings. Since most historical trips usually pass this segment in a much shorter time[6], the predicted arrival time to the bus stop is much smaller than the actual arrival time. Therefore, the large error value is generated.

– The scatter plot shows the absolute prediction errors only, not whether the bus arrived earlier or later than predicted. There is a simple reason for this: the prediction errors are symmetric, due to how my algorithm constructed is. On average, half of the errors are positive, half negative. Thus, it is enough to plot only the absolute errors and not lose any information.

All prediction points are shown on one plot. Hence, the scatter plot can be used as a representative summary of what the algorithm predicts.

(iii) **The table of errors**. Traffic is different at a different time of the day. I split the day into 4 disjoint time intervals:

1. 8am–10am. The morning rush hour.
2. 10am–5pm. The working hours.
2. 5pm–8pm. The evening rush hour.
3. 8pm–8am. I call all other hours as the 'night' time.

[6] e.g. 30s

I evaluate the prediction for each of the time intervals separately. The **table of errors** shows the aggregated results[7]:

Route 1046531-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	196	137	193	181
Oakington	ATT	1979	1755	1797	1740
From stop nr. 2	MAE	169	129	152	137
Histon & Impington	ATT	1722	1537	1549	1483
From stop nr. 3	MAE	160	112	145	118
Regional College	ATT	1486	1342	1346	1281
From stop nr. 4	MAE	148	104	125	109
Science Park	ATT	1379	1238	1247	1186
From stop nr. 5	MAE	99	93	92	90
Central Cambridge	ATT	745	715	684	656
From stop nr. 6	MAE	87	79	92	87
Regent Street	ATT	551	500	505	487
Last stop: Railway Station		138 trips used	198 trips used	57 trips used	165 trips used

The table of errors shows the different prediction errors for different times of the day. It is useful for those who are interested in a particular route and want to check the extent up to which the prediction system can be trusted for that route. For example, suppose I want to take a morning train to London. I can look at the morning column of this table and see that on average it takes 745s. to reach the train station from the Central Cambridge. I can also see that an algorithm on average predicts the arrival time to the train station with an error of 99s.

Note that the average travel time, as well as the mean absolute prediction error values, are larger for the first and third intervals of time. Hence, for this route the traffic is more congested and less predictable during the rush hours.

4.2.2 Evaluation on more Routes

The fact that an algorithm works on a single route does not neccessary imply that it works on other routes. Hence, I evaluate the algorithm on 10 other routes. The detailed results are given in the appendix B.1.

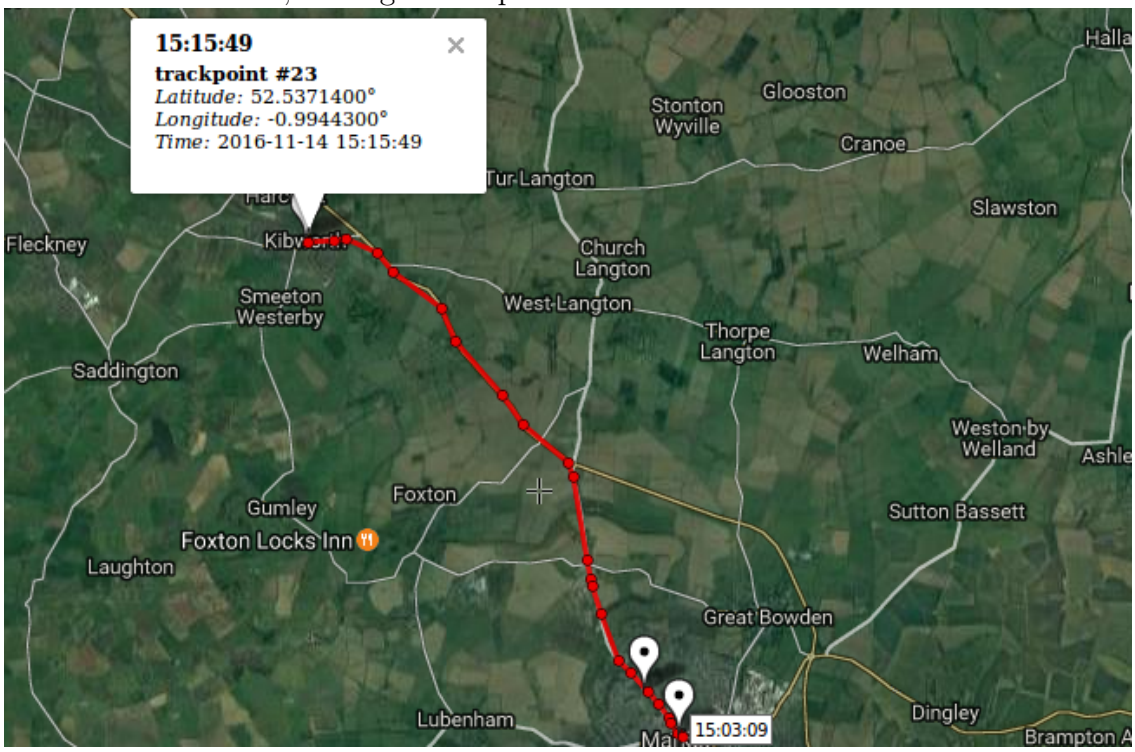
The 10 routes I have chosen are the popular ones. For each route I found more than 100 historical trips passing through all the route's stops. The tables of errors given in the appendix B.1 can serve as a reference point. Anyone writing a new prediction algorithm can check whether it gives lower errors than those mentioned in the appendix.

The Best Route

Route 1056213-20150614-20151223		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	77	67	79	97
The Square	ATT	782	721	742	753
From stop nr. 2	MAE	62	58	43	72
Police Station	ATT	637	595	616	626
Last stop: The Square		7 trips used	62 trips used	31 trips used	48 trips used

[7] As before, I predict the arrival time to the **last** stop starting from each of the other stops

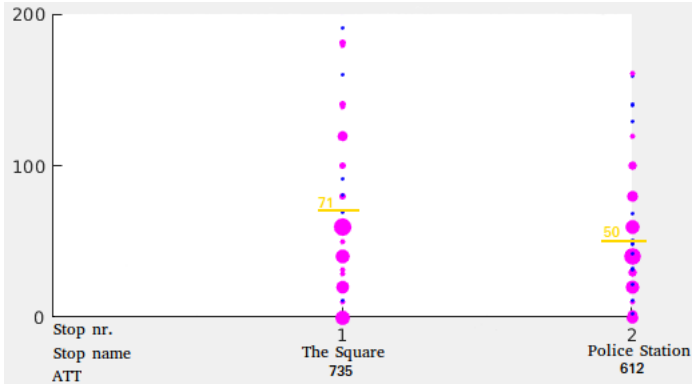
Out of the 10 routes, the algorithm performs best on the route 1056213-20150614-20152323:



For many time intervals, the algorithm predicts the arrival time with an average error less than 10%.

I believe, a long highway path is the reason, why an algorithm successful for this route is. Firstly, highways have a feature that they they are less connected with other roads[8]. Other roads have a relatively little affect on the highway traffic. Hence, only historical trips of the buses travelling along this route are enough to generate reasonable predictions[9]. Secondly, the bus speed on the highway does not vary much. For example, if for the last 5 minutes the bus speed was 60mph., it is likely it will keep travelling at roughly the same speed[10]. This second feature of a highway allows making a good use of the Optimisation nr. 2. The optimisation predicts the arrival time using only those historical trips that have travelled a certain distance using the same amount of time[11] as the current trip. Since the travel speed on a highway does not vary much, these historical trips will give an accurate information how quickly a bus might reach the bus stop.

Finally, I include the scatter plot. As before, many prediction errors are below the MAE value:



[8]Compare a highway with a street in a city. The latter usually has many intersections with other streets. The former has only a few connected roads to go in and out of the highway.

[9]And the implementation of my algorithm uses only historical trips following this route.

[10]Contrast this with the case when a bus is on the street in a city. The bus speed will fluctuate a lot due to traffic lights, other cars causing a congestion, pedestrians crossing the street.

[11] $\Longrightarrow$ the same travel speed.

The Worst Route

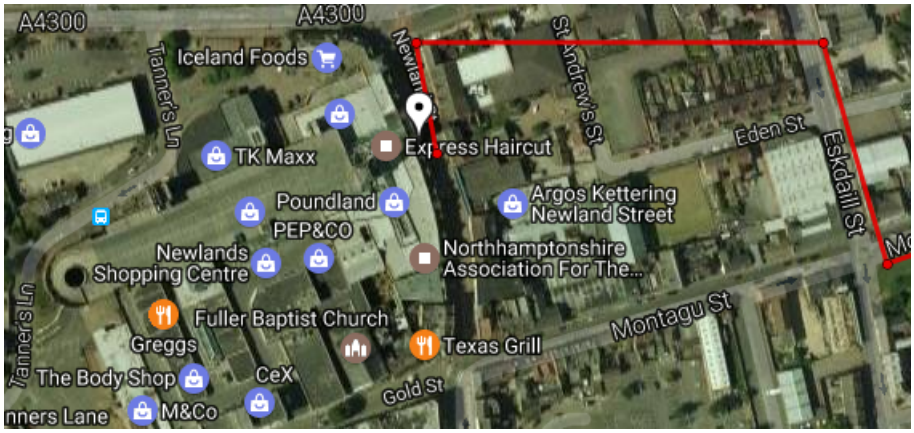
The algorithm performs poorly on the route 999024-20150526-20150830:

Route 999024-20150526-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	129	141	134	136
Newland Centre	ATT	648	666	633	603
From stop nr. 2	MAE	54	40	39	31
Bonham Court	ATT	297	303	273	266
From stop nr. 3	MAE	41	33	29	23
Havelock Street	ATT	222	232	202	205
From stop nr. 4	MAE	34	31	27	24
Lobelia Road	ATT	173	188	163	158
From stop nr. 5	MAE	25	26	26	35
Walnut Crescent	ATT	103	110	97	103
		25 trips used	166 trips used	23 trips used	10 trips used

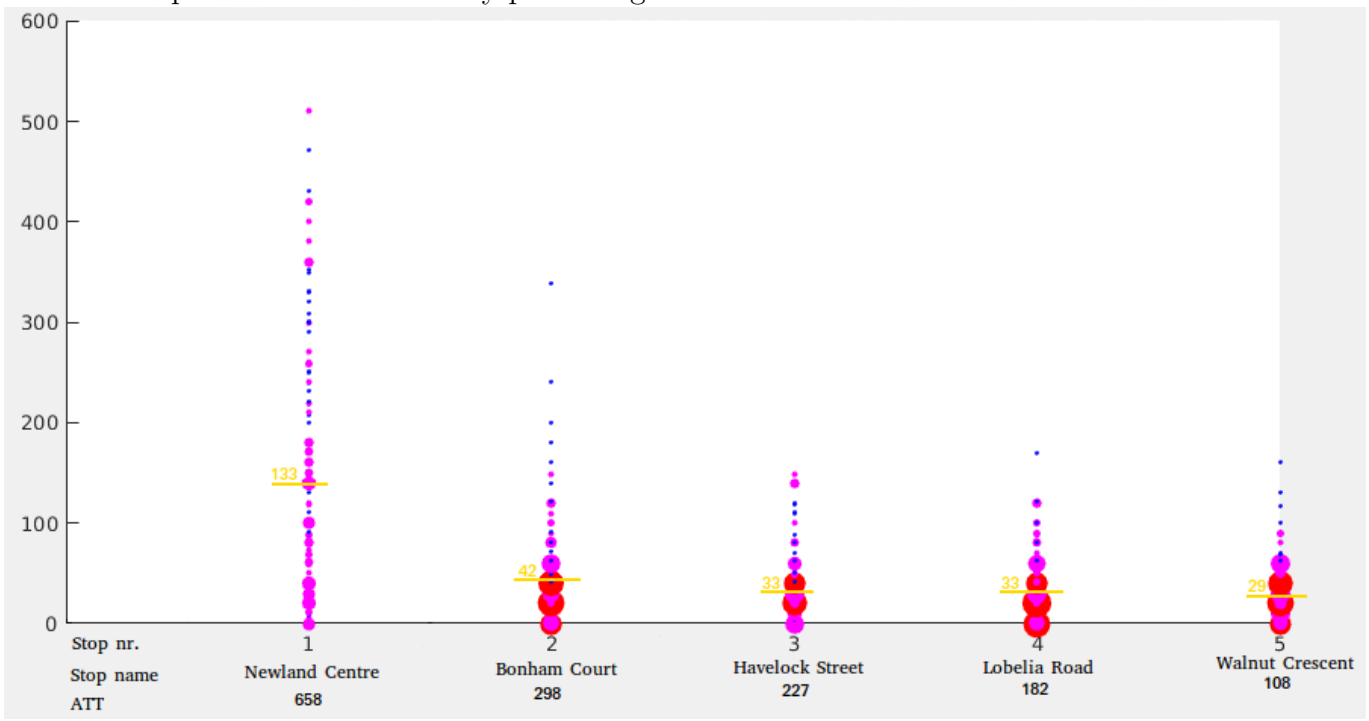
The poor performance is due to the following reasons. Firstly, there are many bus stops a bus frequently[12] stops at. The amount of time a bus is standing at one stop affects its arrival time to the future stop, but the algorithm does not explicitly deal with the bus standing time.

[12]Every 2min. or so.

Secondly, the route includes the crowded city places, such as the shopping centre:



The scatter plot shows the difficulty predicting the arrival time in the crowded areas:



One can see a big difference between the prediction results from the first stop (where the crowded place is) and from the other stops. The prediction errors predicting from the first stop are not concentrated below the MAE value. Instead, the errors are spread out accross the whole range of y values. After the bus leaves the crowded area, the prediction errors become concentrated below the MAE value, which is the usual characteristic of my prediction algorithm.

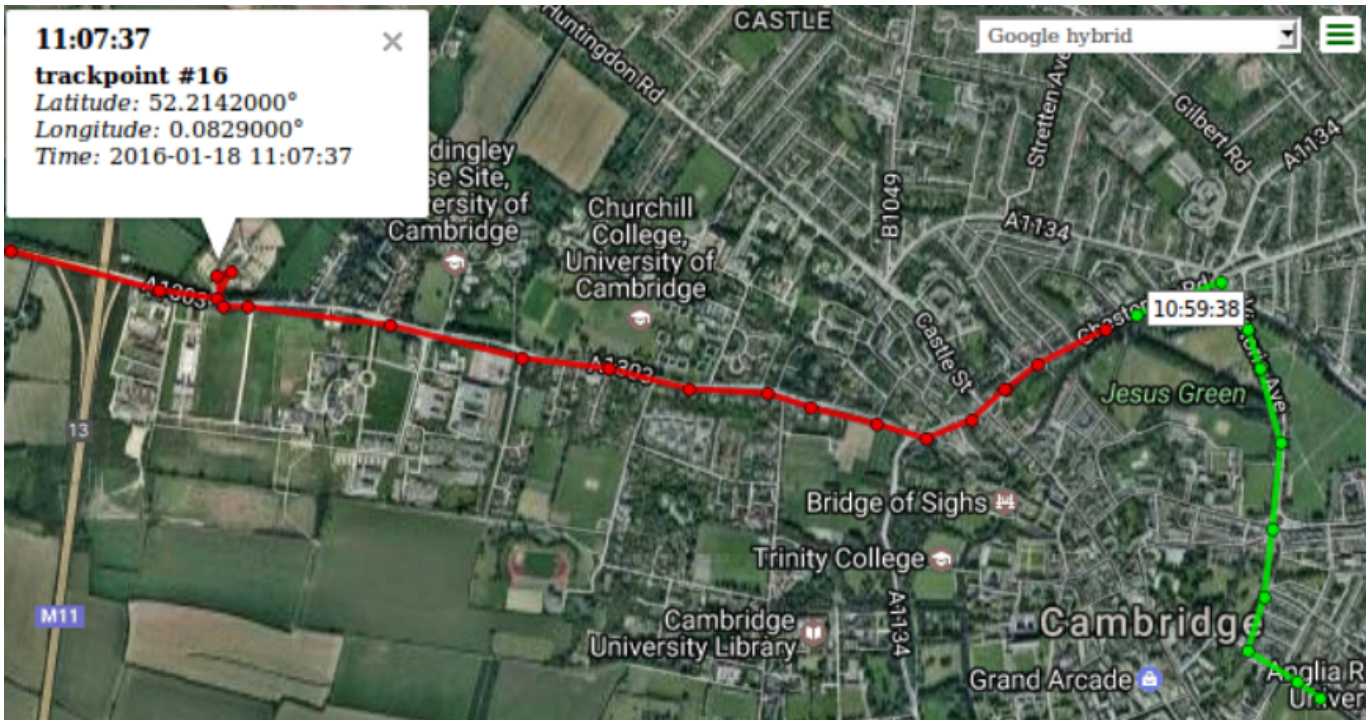
I make the following conclusions:

- My algorithm predicts the arrival time well for buses that are travelling from one city to another and are using highways for a long time.
- My algorithm may perform worse when predicting the arrival time in cities.

4.2.3 Different Systems Comparison

Comparing Optimisations

I first run the non-optimised algorithm to predict when the bus will reach the Madingley Park starting from the Chesterton Road:



Training set having 30 trips results in $MAE_0 = 105$s.[13]

I now apply the optimisation nr. 1 on the same test set.[14] Looking at the detailed results in the appendix, one can see that the optimisation nr. 1 gives a big improvement for certain trips (e.g. the prediction error drops from 380s. to 40s.) and leaves everything else unchanged. Overall, this optimisation reduces the average prediction error to $MAE_1 = 88$s.

Finally, I add the second optimisation to predict the arrival time.[15] Both the 1st and 2nd optimisations together further reduce the average prediction error to $MAE_{1,2} = 69$s.

Hence, I conclude that for this route the optimisations are worth to be applied.

[13] detailed evaluation results are given in the appendix

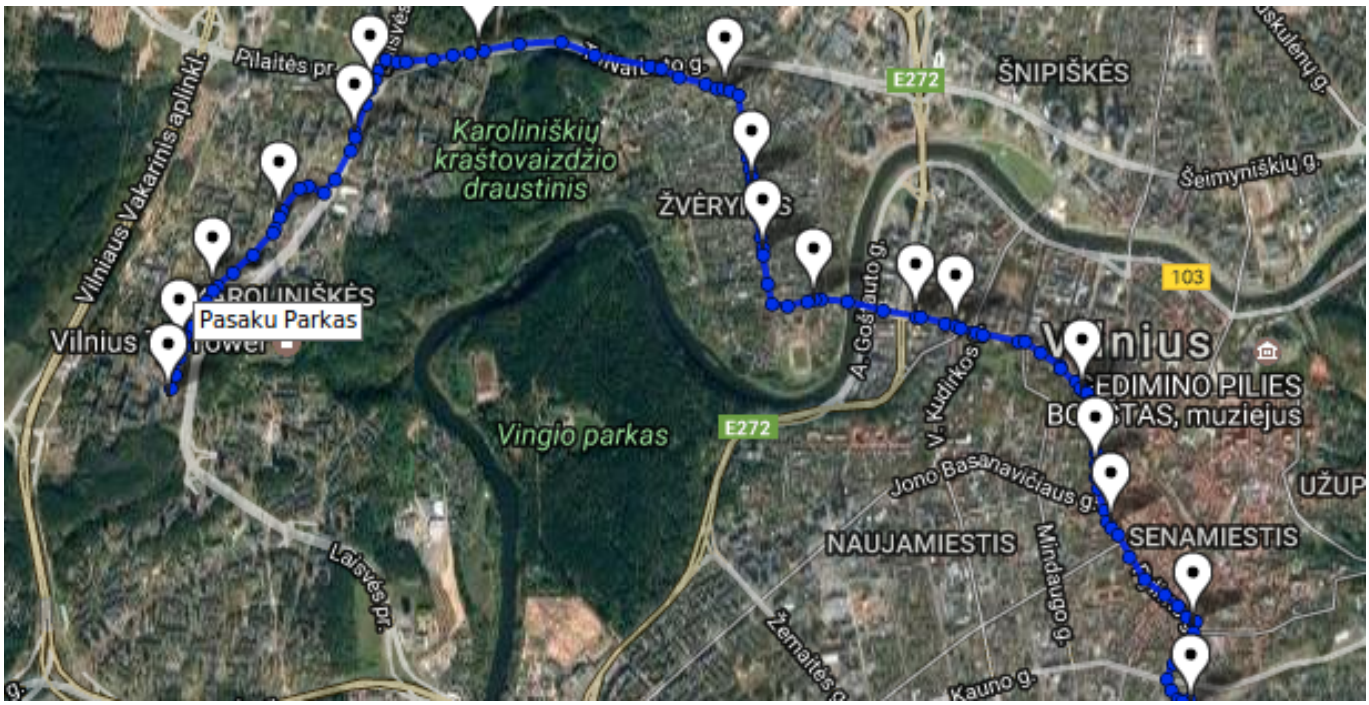
[14] Recall that the first optimisation predicts the arrival time from the recent trips only. If there are no recent trips, the basic approach is used.

[15] The second optimisation takes into account only equally congested historical trips.

Comparison with the Timetable

The prediction system is useful if for a particular route it is more accurate than the timetable. A company named Traffi, which targets the same arrival time prediction problem as me, agreed to share their traffic data[16].

The Traffi data shows when the bus has arrived at the stop **and also** when the bus was expected to arrive according to the timetable. Hence, this data easily allows to my prediction algorithm performance with the timetable. Assume the arrival time prediction is what the timetable announces. Thus, one can view the timetable as just an another prediction algorithm. Then the MAE values for such an algorithm are computed as usual. I compare the MAE values of the 2 algorithms. Thus, I have chosen one of the routes from their dataset and evaluated my algorithm on that route.



The route is from my home city Vilnius. It has 18 bus stops and I predict the arrival time to the last stop 'Pasaku Parkas' from each of the other stops:

The evaluation results are shown on the next page. Two tables of errors are merged into one and presented there. The left number of each coloured cell shows the average prediction error of my algorithm. The right number of each coloured cell shows the average prediction error of the timetable.

Note that for each column the right numbers are all the same. This is because the arrival time announced by the timetable does not change while the bus is moving.

Only 4 MAE values out of 68 are larger for my prediction algorithm. At the beginning of the route, the bus is far away from the last stop, and I have a little amount of recent information how it travels. That is the reason why the prediction errors for the first stops might be worse than what the timetable announces. However, starting from the later stops, the prediction algorithm outperforms the timetable.

[16] I was given the November 2016 GPS data.

		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	103/99	179/180	103/105	105/102
Stotis	ATT	1905	2035	1870	1877
From stop nr. 2	MAE	105/99	166/180	96/105	105/102
Rudininku	ATT	1722	1860	1716	1714
From stop nr. 3	MAE	86/99	155/180	94/105	95/102
Traku	ATT	1563	1682	1556	1551
From stop nr. 4	MAE	88/99	148/180	94/105	85/102
Reformatu	ATT	1476	1585	1465	1463
From stop nr. 5	MAE	84/99	133/180	93/105	88/102
Islandijos	ATT	1345	1442	1324	1333
From stop nr. 6	MAE	79/99	122/180	88/105	85/102
Pamenkalnio	ATT	1151	1242	1149	1158
From stop nr. 7	MAE	78/99	118/180	81/105	82/102
Jokubo Jasinskio	ATT	1099	1190	1098	1108
From stop nr. 8	MAE	70/99	116/180	76/105	81/102
Liubarto Tiltas	ATT	1023	1112	1021	1029
From stop nr. 9	MAE	66/99	103/180	73/105	77/102
Kestucio	ATT	930	1007	926	937
From stop nr. 10	MAE	63/99	89/180	70/105	74/102
Latviu	ATT	871	934	868	881
From stop nr. 11	MAE	56/99	76/180	69/105	71/102
Seliu	ATT	777	823	782	797
From stop nr. 12	MAE	44/99	57/180	58/105	57/102
Teodoro Narburto	ATT	557	593	575	583
From stop nr. 13	MAE	38/99	48/180	50/105	49/102
Sietyno	ATT	428	435	413	419
From stop nr. 14	MAE	39/99	42/180	47/105	45/102
Laisves Prospektas	ATT	375	380	361	367
From stop nr. 15	MAE	30/99	33/180	39/105	32/102
Karoliniskiu Poliklinika	ATT	242	239	233	229
From stop nr. 16	MAE	22/99	26/180	34/105	27/102
Ugniagiesiu	ATT	152	154	152	147
From stop nr. 17	MAE	17/99	22/180	25/105	20/102
Atminties	ATT	73	77	76	72

4.3 Real-time System Evaluation (Optional)

Suppose for a particular bus the real time system detects the route r it follows. When the bus arrives at the b-th bus stop, the system predicts its arrival time $t_{predicted}$ to the last stop of that route. The current time $t_{current}$ and the predicted time are recorded. When the bus arrives at the last stop, the system records its actual arrival time t_{actual}. Thus, the system saves as many tuples

$$(r, b, t_{current}, t_{predicted}, t_{actual})$$

as possible. When I have enough tuples for a particular route r, I can evaluate how well my system performs for that route.

I present here the evaluation of the route 1051308-20150531-22000909. The route has 10 stops. I collected 38 buses passing through this route in 3 days. I made the arrival time prediction to the last stop. The results are as follows:

Route 1028069-20150526-20150830		All day time	
From stop nr. 2	MAE	39	1 prediction
St Pauls Square	ATT	801	
From stop nr. 3	MAE	140	29 predictions
Kingsway Link	ATT	576	
From stop nr. 4	MAE	200	30 predictions
Borough Hall	ATT	520	
From stop nr. 5	MAE	112	30 predictions
Bedford Hospital	ATT	417	
From stop nr. 6	MAE	67	34 predictions
Whitbread Avenue	ATT	314	
From stop nr. 7	MAE	45	38 predictions
The Keep	ATT	254	
From stop nr. 8	MAE	31	38 predictions
Spring Road	ATT	180	
From stop nr. 9	MAE	22	38 predictions
Margetts Road	ATT	113	

Chapter 5

Conclusion

I implemented an algorithm which gives sensible prediction results. The algorithm itself is simple, thus anyone coming after me to improve the system should not have a trouble to read and understand what has been done so far.

Appendix A

Links

1. Dynamic time warping is an algorithm for measuring similarity between two temporal sequences which may vary in speed. For instance, similarities in walking could be detected using DTW, even if one person was walking faster than another. I used DTW algorithm as a core part to implement my $FollowsPath$ predicate, with err function being used instead of a regular distance function $d(x, y)$. For more info look at

   https://en.wikipedia.org/wiki/Dynamic_time_warping

2. Project code can be found at:

   https://github.com/ml693/bus

Appendix B

Detailed Evaluation Results

B.1 Multiple Routes Evaluation

Route 1046521-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	203	137	107	169
Station Road	ATT	2502	2334	2261	2294
From stop nr. 2	MAE	213	121	114	153
St Ives P & R	ATT	2388	2225	2157	2192
From stop nr. 3	MAE	162	107	111	137
Fen Drayton Lakes	ATT	2116	1948	1964	1974
From stop nr. 4	MAE	139	106	104	140
Swavesey	ATT	1968	1810	1828	1830
From stop nr. 5	MAE	111	107	107	145
Longstanton P & R	ATT	1703	1560	1587	1567
From stop nr. 6	MAE	134	74	100	118
Oakington	ATT	1441	1326	1349	1302
From stop nr. 7	MAE	114	72	107	111
Histon & Impington	ATT	1178	1091	1102	1053
From stop nr. 8	MAE	74	69	73	110
Regional College	ATT	954	895	899	841
From stop nr. 9	MAE	68	66	83	102
Science Park	ATT	841	794	803	746
From stop nr. 10	MAE	32	29	34	34
Central Cambridge	ATT	225	249	233	209
		49 trips used	39 trips used	17 trips used	55 trips used

Route 999024-20150526-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	129	141	134	136
Newland Centre	ATT	648	666	633	603
From stop nr. 2	MAE	54	40	39	31
Bonham Court	ATT	297	303	273	266
From stop nr. 3	MAE	41	33	29	23
Havelock Street	ATT	222	232	202	205
From stop nr. 4	MAE	34	31	27	24
Lobelia Road	ATT	173	188	163	158
From stop nr. 5	MAE	25	26	26	35
Walnut Crescent	ATT	103	110	97	103
		25 trips used	166 trips used	23 trips used	10 trips used

Route 1028069-20150526-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	242	123	103	158
Luton Stn Stand 10	ATT	2502	2334	2261	2294
From stop nr. 2	MAE	173	110	105	143
Clifton Rd West	ATT	2388	2225	2157	2192
From stop nr. 3	MAE	129	117	113	177
Stanton Rd West	ATT	2116	1948	1964	1974
From stop nr. 4	MAE	130	114	104	143
White Lion West	ATT	1968	1810	1828	1830
From stop nr. 5	MAE	118	104	95	142
CB College	ATT	1703	1560	1587	1567
From stop nr. 6	MAE	121	78	86	110
Stop P - Court Dr	ATT	1441	1326	1349	1302
From stop nr. 7	MAE	97	69	78	104
Stop M - High St	ATT	1178	1091	1102	1053
		30 trips used	65 trips used	39 trips used	28 trips used

Route 1053969-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	428	254	342	661
Luton Stn Stand 11	ATT	3848	3257	3469	3847
From stop nr. 2	MAE	363	240	296	718
Galaxy Centre	ATT	3667	3359	3331	3661
From stop nr. 3	MAE	273	211	127	501
Halfway Avenue	ATT	2939	2658	2579	2933
From stop nr. 4	MAE	169	145	107	221
Brinklow Roundabout	ATT	1204	1173	1118	1182
From stop nr. 5	MAE	107	109	94	138
The Point	ATT	568	595	593	562
Last stop: Central Railway		19 trips used	56 trips used	19 trips used	41 trips used

Route 1030740-20150601-20151231		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	108	103	152	145
Stop G - Church St	ATT	826	805	856	870
From stop nr. 2	MAE	64	84	105	70
Stanton Rd East	ATT	466	485	491	495
From stop nr. 3	MAE	44	67	72	58
Clifton Rd East	ATT	222	246	249	250
From stop nr. 4	MAE	34	60	55	44
Shuttle Link	ATT	89	112	106	113
		75 trips used	186 trips used	66 trips used	97 trips used

Route 830638-20150526-22000909		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	99	62	61	79
33 Park Drive	ATT	383	298	322	416
From stop nr. 2	MAE	96	61	61	89
South Oval Shops	ATT	374	284	297	397
From stop nr. 3	MAE	64	39	41	23
Witham Way	ATT	191	176	180	174
From stop nr. 4	MAE	36	33	48	23
Woodside Walk	ATT	128	126	132	128
Last stop: Witham Green		37 trips used	177 trips used	38 trips used	49 trips used

Route 1039292-20150531-22000909		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	192	132	206	220
Trumpington P & R	ATT	824	871	751	863
From stop nr. 2	MAE	93	151	142	56
Porson Road	ATT	409	533	415	363
From stop nr. 3	MAE	81	121	93	42
Nuffield Hospital	ATT	303	388	323	272
From stop nr. 4	MAE	53	47	43	32
Leys School	ATT	186	195	188	166
Last stop: Pembroke Street		43 trips used	79 trips used	14 trips used	27 trips used

Route 1001579-20150601-20150724		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	207	220	331	179
North Gate Bay 11	ATT	1223	1325	1486	1185
From stop nr. 2	MAE	134	104	173	110
Abington Square	ATT	929	992	1014	889
Last stop: Northampton College		189 trips used	805 trips used	211 trips used	279 trips used

Route 1056213-20150614-20151223		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	77	67	79	97
The Square	ATT	782	721	742	753
From stop nr. 2	MAE	62	58	43	72
Police Station	ATT	637	595	616	626
Last stop: The Square		7 trips used	62 trips used	31 trips used	48 trips used

Route 1002879-20150601-20150724		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	98	83	147	71
School for Boys	ATT	601	641	681	533
From stop nr. 2	MAE	121	78	160	75
Lutterworth Road	ATT	556	595	643	502
From stop nr. 3	MAE	70	70	177	71
St Andrews Hospital	ATT	493	541	580	451
From stop nr. 4	MAE	54	52	70	29
Palmerston Road	ATT	345	412	402	340
From stop nr. 5	MAE	60	51	105	25
Cliftonville	ATT	311	369	359	299
From stop nr. 6	MAE	58	46	94	35
Alexandra Road	ATT	249	302	284	225
Last stop: Northampton Draper		14 trips used	59 trips used	23 trips used	20 trips used

B.2 Predictions for Madingley Park

Results without optimisations:

day18-bus14365-subtrip0 started at 2016-01-18 13:45:46, arrived at 2016-01-18 13:52:06, prediction error is -80
day18-bus14368-subtrip2 started at 2016-01-18 21:14:38, arrived at 2016-01-18 21:21:18, prediction error is -60
day18-bus14366-subtrip0 started at 2016-01-18 11:46:35, arrived at 2016-01-18 11:55:36, prediction error is 81
day18-bus14369-subtrip1 started at 2016-01-18 17:46:33, arrived at 2016-01-18 17:55:13, prediction error is 59
day18-bus403-subtrip0 started at 2016-01-18 11:23:07, arrived at 2016-01-18 11:28:07, prediction error is -60
day18-bus14374-subtrip0 started at 2016-01-18 16:15:35, arrived at 2016-01-18 16:22:35, prediction error is 19
day18-bus14361-subtrip0 started at 2016-01-18 10:16:14, arrived at 2016-01-18 10:28:34, prediction error is 340
day18-bus14370-subtrip0 started at 2016-01-18 10:30:47, arrived at 2016-01-18 10:44:47, prediction error is **380**
day18-bus14365-subtrip1 started at 2016-01-18 21:44:37, arrived at 2016-01-18 21:50:17, prediction error is -80
day18-bus407-subtrip1 started at 2016-01-18 10:15:37, arrived at 2016-01-18 10:28:40, prediction error is 334
day18-bus14368-subtrip1 started at 2016-01-18 13:14:46, arrived at 2016-01-18 13:20:06, prediction error is -100
day18-bus14363-subtrip1 started at 2016-01-18 15:46:05, arrived at 2016-01-18 15:53:45, prediction error is -20
day18-bus14375-subtrip0 started at 2016-01-18 15:14:50, arrived at 2016-01-18 15:21:10, prediction error is -69
day18-bus401-subtrip2 started at 2016-01-18 11:46:37, arrived at 2016-01-18 11:55:38, prediction error is 81
day18-bus14370-subtrip3 started at 2016-01-18 19:30:41, arrived at 2016-01-18 19:38:42, prediction error is 121
day18-bus410-subtrip0 started at 2016-01-18 16:15:38, arrived at 2016-01-18 16:22:38, prediction error is 20
day18-bus402-subtrip0 started at 2016-01-18 10:30:38, arrived at 2016-01-18 10:44:37, prediction error is **379**
day18-bus14373-subtrip0 started at 2016-01-18 16:46:02, arrived at 2016-01-18 16:53:42, prediction error is -1
day18-bus14372-subtrip1 started at 2016-01-18 14:14:19, arrived at 2016-01-18 14:20:19, prediction error is -120
day18-bus14367-subtrip2 started at 2016-01-19 00:35:49, arrived at 2016-01-19 00:40:49, prediction error is -160
day18-bus14378-subtrip0 started at 2016-01-18 11:00:21, arrived at 2016-01-18 11:07:41, prediction error is 20
day18-bus402-subtrip1 started at 2016-01-18 19:30:08, arrived at 2016-01-18 19:38:37, prediction error is 89
day18-bus400-subtrip0 started at 2016-01-18 10:59:38, arrived at 2016-01-18 11:07:37, prediction error is 29
day18-bus14087-subtrip0 started at 2016-01-18 09:30:34, arrived at 2016-01-18 09:36:34, prediction error is -120
day18-bus409-subtrip0 started at 2016-01-18 08:25:38, arrived at 2016-01-18 08:32:09, prediction error is -29
day18-bus14371-subtrip1 started at 2016-01-18 08:54:47, arrived at 2016-01-18 09:01:07, prediction error is -70
day18-bus14072-subtrip3 started at 2016-01-18 20:51:07, arrived at 2016-01-18 20:57:47, prediction error is -79
day18-bus14072-subtrip1 started at 2016-01-18 12:45:18, arrived at 2016-01-18 12:53:18, prediction error is 20
day18-bus14362-subtrip1 started at 2016-01-18 14:45:16, arrived at 2016-01-18 14:50:36, prediction error is -81
day18-bus405-subtrip2 started at 2016-01-18 13:46:07, arrived at 2016-01-18 13:52:08, prediction error is -59

Results with the optimisation nr. 1:

day18-bus14365-subtrip0 started at 2016-01-18 13:45:46, arrived at 2016-01-18 13:52:06, prediction error is -80
day18-bus14368-subtrip2 started at 2016-01-18 21:14:38, arrived at 2016-01-18 21:21:18, prediction error is 0
day18-bus14366-subtrip0 started at 2016-01-18 11:46:35, arrived at 2016-01-18 11:55:36, prediction error is 81
day18-bus14369-subtrip1 started at 2016-01-18 17:46:33, arrived at 2016-01-18 17:55:13, prediction error is 59
day18-bus403-subtrip0 started at 2016-01-18 11:23:07, arrived at 2016-01-18 11:28:07, prediction error is -60
day18-bus14374-subtrip0 started at 2016-01-18 16:15:35,arrived at 2016-01-18 16:22:35, prediction error is 19
day18-bus14361-subtrip0 started at 2016-01-18 10:16:14, arrived at 2016-01-18 10:28:34, prediction error is 340
day18-bus14370-subtrip0 started at 2016-01-18 10:30:47, arrived at 2016-01-18 10:44:47, prediction error is **40**
day18-bus14365-subtrip1 started at 2016-01-18 21:44:37, arrived at 2016-01-18 21:50:17, prediction error is -60
day18-bus407-subtrip1 started at 2016-01-18 10:15:37, arrived at 2016-01-18 10:28:40, prediction error is 334
day18-bus14368-subtrip1 started at 2016-01-18 13:14:46, arrived at 2016-01-18 13:20:06, prediction error is -100
day18-bus14363-subtrip1 started at 2016-01-18 15:46:05, arrived at 2016-01-18 15:53:45, prediction error is -20
day18-bus14375-subtrip0 started at 2016-01-18 15:14:50, arrived at 2016-01-18 15:21:10, prediction error is -69
day18-bus401-subtrip2 started at 2016-01-18 11:46:37, arrived at 2016-01-18 11:55:38, prediction error is 81
day18-bus14370-subtrip3 started at 2016-01-18 19:30:41, arrived at 2016-01-18 19:38:42, prediction error is 121
day18-bus410-subtrip0 started at 2016-01-18 16:15:38, arrived at 2016-01-18 16:22:38, prediction error is 20
day18-bus402-subtrip0 started at 2016-01-18 10:30:38, arrived at 2016-01-18 10:44:37, prediction error is **39**
day18-bus14373-subtrip0 started at 2016-01-18 16:46:02, arrived at 2016-01-18 16:53:42, prediction error is -1
day18-bus14372-subtrip1 started at 2016-01-18 14:14:19, arrived at 2016-01-18 14:20:19, prediction error is -31
day18-bus14367-subtrip2 started at 2016-01-19 00:35:49, arrived at 2016-01-19 00:40:49, prediction error is -160
day18-bus14378-subtrip0 started at 2016-01-18 11:00:21, arrived at 2016-01-18 11:07:41, prediction error is -200
day18-bus402-subtrip1 started at 2016-01-18 19:30:08, arrived at 2016-01-18 19:38:37, prediction error is 89
day18-bus400-subtrip0 started at 2016-01-18 10:59:38, arrived at 2016-01-18 11:07:37, prediction error is -241
day18-bus14087-subtrip0 started at 2016-01-18 09:30:34, arrived at 2016-01-18 09:36:34, prediction error is -120
day18-bus409-subtrip0 started at 2016-01-18 08:25:38, arrived at 2016-01-18 08:32:09, prediction error is -29
day18-bus14371-subtrip1 started at 2016-01-18 08:54:47, arrived at 2016-01-18 09:01:07, prediction error is -42
day18-bus14072-subtrip3 started at 2016-01-18 20:51:07, arrived at 2016-01-18 20:57:47, prediction error is -79
day18-bus14072-subtrip1 started at 2016-01-18 12:45:18, arrived at 2016-01-18 12:53:18, prediction error is 20
day18-bus14362-subtrip1 started at 2016-01-18 14:45:16, arrived at 2016-01-18 14:50:36, prediction error is -81
day18-bus405-subtrip2 started at 2016-01-18 13:46:07, arrived at 2016-01-18 13:52:08, prediction error is 41

Results with the optimisations nr. 1 and nr. 2:

day18-bus14365-subtrip0 started at 2016-01-18 13:45:46, arrived at 2016-01-18 13:52:06, prediction error is -100
day18-bus14368-subtrip2 started at 2016-01-18 21:14:38, arrived at 2016-01-18 21:21:18, prediction error is -20
day18-bus14366-subtrip0 started at 2016-01-18 11:46:35, arrived at 2016-01-18 11:55:36, prediction error is 41
day18-bus14369-subtrip1 started at 2016-01-18 17:46:33, arrived at 2016-01-18 17:55:13, prediction error is 40
day18-bus403-subtrip0 started at 2016-01-18 11:23:07, arrived at 2016-01-18 11:28:07, prediction error is -59
day18-bus14374-subtrip0 started at 2016-01-18 16:15:35, arrived at 2016-01-18 16:22:35, prediction error is 0
day18-bus14361-subtrip0 started at 2016-01-18 10:16:14,arrived at 2016-01-18 10:28:34, prediction error is 280
day18-bus14370-subtrip0 started at 2016-01-18 10:30:47, arrived at 2016-01-18 10:44:47, prediction error is 40
day18-bus14365-subtrip1 started at 2016-01-18 21:44:37, arrived at 2016-01-18 21:50:17, prediction error is -51
day18-bus407-subtrip1 started at 2016-01-18 10:15:37, arrived at 2016-01-18 10:28:40, prediction error is 302
day18-bus14368-subtrip1 started at 2016-01-18 13:14:46, arrived at 2016-01-18 13:20:06, prediction error is -100
day18-bus14363-subtrip1 started at 2016-01-18 15:46:05, arrived at 2016-01-18 15:53:45, prediction error is -20
day18-bus14375-subtrip0 started at 2016-01-18 15:14:50, arrived at 2016-01-18 15:21:10, prediction error is -42
day18-bus401-subtrip2 started at 2016-01-18 11:46:37, arrived at 2016-01-18 11:55:38, prediction error is 60

day18-bus14370-subtrip3 started at 2016-01-18 19:30:41, arrived at 2016-01-18 19:38:42, prediction error is 122
day18-bus410-subtrip0 started at 2016-01-18 16:15:38, arrived at 2016-01-18 16:22:38, prediction error is 20
day18-bus402-subtrip0 started at 2016-01-18 10:30:38, arrived at 2016-01-18 10:44:37, prediction error is 39
day18-bus14373-subtrip0 started at 2016-01-18 16:46:02, arrived at 2016-01-18 16:53:42, prediction error is -20
day18-bus14372-subtrip1 started at 2016-01-18 14:14:19, arrived at 2016-01-18 14:20:19, prediction error is -31
day18-bus14367-subtrip2 started at 2016-01-19 00:35:49, arrived at 2016-01-19 00:40:49, prediction error is -180
day18-bus14378-subtrip0 started at 2016-01-18 11:00:21, arrived at 2016-01-18 11:07:41, prediction error is 20
day18-bus402-subtrip1 started at 2016-01-18 19:30:08, arrived at 2016-01-18 19:38:37, prediction error is 108
day18-bus400-subtrip0 started at 2016-01-18 10:59:38, arrived at 2016-01-18 11:07:37, prediction error is 29
day18-bus14087-subtrip0 started at 2016-01-18 09:30:34, arrived at 2016-01-18 09:36:34, prediction error is -60
day18-bus409-subtrip0 started at 2016-01-18 08:25:38, arrived at 2016-01-18 08:32:09, prediction error is -9
day18-bus14371-subtrip1 started at 2016-01-18 08:54:47, arrived at 2016-01-18 09:01:07, prediction error is -42
day18-bus14072-subtrip3 started at 2016-01-18 20:51:07, arrived at 2016-01-18 20:57:47, prediction error is -81
day18-bus14072-subtrip1 started at 2016-01-18 12:45:18, arrived at 2016-01-18 12:53:18, prediction error is -1
day18-bus14362-subtrip1 started at 2016-01-18 14:45:16, arrived at 2016-01-18 14:50:36, prediction error is -80
day18-bus405-subtrip2 started at 2016-01-18 13:46:07, arrived at 2016-01-18 13:52:08, prediction error is -99

Appendix C

Project Proposal

(see the next page)

Bus Arrival Time Prediction

Computer Science Tripos - Part II - Project Proposal

Marius Latinis (ml693@cam.ac.uk)

Project Originator: Dr. Richard Mortier
Project Supervisor: Dr. Richard Mortier
Director of Studies: Prof. Ian Leslie
Project Overseers: Dr. Markus Kuhn & Prof. Peter Sewell
Document Creation Date: 13 October 2016

Introduction

In European countries public buses is a popular form of transportation. However, buses often arrive later than announced at the stop. This annoys passengers and makes them complain. Wouldn't it be nice to have a good algorithm that predicts the bus arrival times precisely? This project targets such question.

Starting Point

Communication with buses streaming GPS data is already established. A file showing the GPS snapshot of 2 months (June and July) will be given to me as a starting data to work with. Real time bus GPS data will be presented for the optional extension. According to Dr. Lewis, the prepared and convenient bus GPS data files to work with are written in JSON format. They are roughly in one to one correspondence with the GTFS-realtime files, which are the files actually sent from the buses. Hence, I will work with JSON format as it is more readable and easier to process. That's the starting point data.

Work to be done

The project breaks down into the following parts:

1. Prepare maps for processing. A graph model of the map is considered in this project. The streets will be represented as edges with vertices being their intersections. One needs to **find** a map that contains streets covering the bus stops of consideration.

2. Prepare bus GPS data for processing. Bus GPS data also has to be prepared. The short description of what the bus data looks like according to Dr. Lewis is as follows:

> *"Buses announce their location data using GTFS-realtime format once every **30s**. GTFS-realtime format is then converted to a more readable JSON file. Each file contains information (current GPS location, unique bus identifier, bus route identifier and more) of around 1000 buses. "*

We hope that ***30s*** granularity will be enough to estimate the time it takes for a bus to travel from one intersection to the next intersection. Since it is unlikely that the bus will announce its data exactly at the intersection, we will have to interpolate the data.

The **core** project's part is to work only with buses in Cambridge and its neighbourhood. However, there might exist buses in the data that are out of the project's scope (i.e. a bus travelling to London). These will have to be filtered out.

3. Prediction algorithm implementation. Based on the data extracted in paragraphs (1) and (2), the algorithm has to inform when the next bus arrives to which stop. Suppose we are aiming to predict when a bus β being in a current location l_1 will reach the location l_n after having travelled through locations l_2, l_3, l_{n-1} in between (location is a vertex in a graph). For each i we ***average*** the most recent time $t_{i,\, i+1}$ taken for other buses to travel from l_i to l_{i+1}. Then the predicted time for β to reach l_n will be a sum $\sum_{i=1}^{n-1} t_{i,\, i+1}$.

Research concerning arrival time prediction has already been done in another paper[1]. The new prediction algorithm will differ in a few ways. Firstly, that paper essentially predicts a single value $t_{1,\, n}$ and outputs it as a result. Instead, we will calculate multiple intermediate values $t_{i,\, i+1}$, each of which will correspond to a predicted time for a bus to travel across one route's edge. We ***hypothesize*** that accurately predicting time for shorter segments and then summing all times up will lead to a better precision (read part 3 how the hypothesis will be tested). Secondly, the paper uses general machine learning models as a tool to predict arrival time. We will use calculations specific to the arrival prediction problem. That will ease the job of refining the algorithm in case it is not working well enough (see the first extension as an example how). It will also make the algorithm's optimisation task easier, as the algorithm itself targets a specific problem, rather than being a general ML tool.

// The pseudocode for the prediction algorithm

For each bus β streaming GPS data:

a) Extract l_1 - the current location of β

b) Look at β route to see the next locations $l_2, \ldots, l_n$

// The (c) part will be implemented following the summation idea above

c) Predict β arrival times $t_2, \ldots, t_n$ for each $l_2, \ldots, l_n$ and store these predictions in a multimap data structure *Bus[*β*] Arrives{(*l_2, t_2*), ... (*l_n, t_n*) }*

4. Prediction algorithm evaluation. Mathematical evaluation will be measured based on how well the algorithm predicts new arrival times. We will compare the prediction results with the actual time after we know when the bus has arrived. We want to use an evaluation metric similar to what's used elsewhere[2] (also known as the MAE_t value):

> *"In the trace-driven study, we [...] compare the predicted arrival time with the actual arrival time of the campus buses to compute the average of the absolute prediction error."*

Various error results are claimed (based on the bus distance to the stop) with ***80s*** being one of the values in that paper. Hence, we set a goal to build a model that would produce results with an error smaller than ***80s***. We will compute the MAE_t with $t_{1,n}$ being replaced by the sum $\sum_{i=1}^{n-1} t_{i,i+1}$. If the new MAE_t value is smaller than ***80s***, we claim that the **hypothesis** and hence the project was successful.

However, just assessing the project based on one number does not look convincing. To improve the assessment we propose to do a more in depth **quantitative evaluation**. Rather than computing a one global MAE_t value, we will also compute the MAE_t for each bus route and for each segment. In addition to that, we will count the total number of buses that arrived more than ***80s*** later, regardless of how late they arrived. All of these numbers will be reported in the progress report and final dissertation. The aim is to provide a few interesting metrics that will allow other researchers to have a choice in case they want to optimise for a particular one.

Finally, one more way to evaluate the project is by analysing how easily the algorithm can be run on a large scale. We will present this analysis in the dissertation by reporting its current complexity (w.r.t. the graph and JSON files size) and giving a clue what would be the sequential algorithm's part if run on multiple machines. That will provide others an idea of whether it is worthful to extend the project. However, **we would like to**

emphasize that scalability is not the main concern of the project, so going into too much detail here would be an overkill.

These four parts together form the core of the project.

Possible extensions

5. Optimise the prediction algorithm from the precision point of view. An advise how to do that was given by the overseer Dr. Markus Kuhn:

> "I would start with first estimating average edge times as a function of time of day and time of week, as load on the road network is usually a quite periodic process. I would then look into a textbook on standard multivariate statistical analysis for algorithm ideas on how to exploit statistical dependencies between spatially and temporally related edge traversal times. For example you could determine covariance matrices that describe how recently measured time intervals correlate with time intervals in the near future on either the same edge or on other nearby edges in your route graph. These averages and correlation matrices could then be used to calculate conditional probability distributions for traversal times in the near future, which could be added to make predictions. If you try to independently estimate the probability distribution of traversal time for each of thousands of edges, you may end up with quite noisy predictions, if you have more parameters to estimate than there are data points. This is where dimensionality-reduction techniques (such as principal component analysis) can become useful, where you try to represent the large vectors of edge traversal times that you try to predict as linear combinations of a much smaller number of base vectors (for principal component analysis, these will be eigenvectors of the covariance matrices), which still represent well the overall traffic situation in e.g. a town, but with far fewer parameters to estimate. Fewer parameters are easier to estimate from limited data, and hence allow faster updates."

6. Turning the static GPS data analysis into the real time processing. If our prediction algorithm gives promising results on the static data, extending it to work with the real time data would allow people to see when their desired bus actually arrives at the stop. This will involve writing a batch processing job that takes GPS files as input and continuously reports new arrival times as output. According to Dr. Lewis, the processing will be similar to the processing of stock transactions. Hence, we will split this task into the learning (i.e. surfing the net of how transactions are processed) and implementation parts.

7. Building a webpage to display prediction results. Given that we have predicted results, it would be nice to show them to the public. That's what the webpage will be used for. The aim is to have one stateless page where all the info is placed. Below is the example of the webpage content,

Station's Name	Next buses to arrive and arrival times
Covent Garden	No. 2 in 4min., No. 1 in 7min.
Leicester Square	No. 1 in 4min., No. 8 in 6min., No. 7 in 10min.

Resources required

The main resource required is "Bus GPS position data via Ian Lewis".

Other resources are my personal laptop running Ubuntu, GitHub, Google Docs, Hermes emailing system, SRCF.

Success Criteria

The project will be a success if I have implemented a working prediction algorithm that achieves ***error*** (*error* = average absolute error of real time prediction values*)* less than ***80s***. In addition to this main error value, we will plot an in depth **quantitative evaluation** statistics. Good results of other numbers will reinforce the main success criterion.

Timetable (planned starting date is 21/10/2011)

- Michaelmas term weeks 3 - 4: find maps open source data and make a graph model of it. A milestone for this part is to have a file containing a nice graph representation of Cambridge bus routes.

- Michaelmas term weeks 5 - 7: write a function which takes a JSON file as an input, extracts every bus with its GPS position, and for each bus finds the closest vertex where the bus currently is. One will also need to filter out those buses that

are present in the file but not in the project's scope (e.g. a bus travelling from Cambridge to London). Furthermore, some data might turn out to be garbage (e.g. bus presence along the wrong route) and this has to be taken into account. As it is important to prepare the main GPS data well, I am therefore giving 3 weeks for this part. This extra time will also help to rethink about the size of the graph once I started using it.

- Michaelmas term week 8: implement an "empty" framework. The framework will take JSON files and Cambridge map as an input and run the fake prediction algorithm on the input. The unimplemented fake part is (roughly) a function that takes a list of numbers as an input (the time it takes for other buses to travel across an edge) and produces a single number as an output (the predicted arrival time). The milestone here is to make sure that the framework itself is running, even if the results it displays are wrong.

- Christmas holidays weeks 1 - 3: replace the fake algorithm by the core prediction function. Evaluate the algorithm by computing the error numbers. If an error is too large, attempt to find a simple refinement of the core algorithm.

- Christmas holidays week 4: gather the best current algorithm statistics and write a progress report. **These results will be given for the midpoint presentation. Note that if everything is OK up to here, that means the core part of the project has been completed.** For the Lent term I am planning to work on extensions and am giving a further timetable:

- Lent weeks 1 - 3: work on optimising the algorithm from the precision point of view. One week can be spent reading about multivariate normal distribution, as it is likely to be used. Second week would be allocated for refining the algorithm. Third week would be spent comparing the new results with the best we have so far.
- Lent week 4: read how the stock data is processed in real time.
- Lent weeks 5 - 7: based on the info read implement the real time processing system.
- Lent week 8: create a webpage to display the results.

- Easter vacation: double check the work done before, write the first dissertation's draft.
- Easter term weeks 1 - 2: create the final dissertation's version.
- Easter term week 3: submit the dissertation.

Literature Appendix

1. Bus arrival time prediction at bus stop with multiple routes - Bin Yu , William H.K. Lam, Mei Lam Tam (http://www.sciencedirect.com/science/article/pii/S0968090X11000155)
2. How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing - Pengfei Zhou, Yuanqing Zheng, Mo Li (http://www.ntu.edu.sg/home/limo/papers/sys012fp.pdf)
3. Bus Based Probe Urban Travel Time Prediction - Wenjing Pu(https://books.google.co.uk/books?id=Tdps9w5jHKEC&pg=PA117&lpg=PA117&dq=multivariate+normal+distribution+for+bus+prediction&source=bl&ots=p5Aqeyo2NO&sig=Ml2XsklcIPBf9ChuR7ZdJLX2Qmc&hl=en&sa=X&ved=0ahUKEwj8soTzuenPAhVFBMAKHcC2CM8Q6AEIMzAD#v=onepage&q&f=false)
4. Applied Multivariate Statistical Analysis - R. Johnson, D. Wichern (https://www.pearsonhighered.com/program/Johnson-Applied-Multivariate-Statistical-Analysis-6th-Edition/PGM274834.html)
5. Multivariate Normal Distribution - Wikipedia (https://en.wikipedia.org/wiki/Multivariate_normal_distribution)