Marius Latinis

Bus Arrival Time Prediction

Computer Science Tripos – Part II

Christ's College

May 4, 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 4, 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. They use the public transport for that. For example, a bus is used 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 disappointing to miss a departing train due to a late bus. Therefore, people are looking for a source of information reliably telling when a vehicle will reach a certain place.

1.2 Timetable Limits

The timetable is one potential source of information:

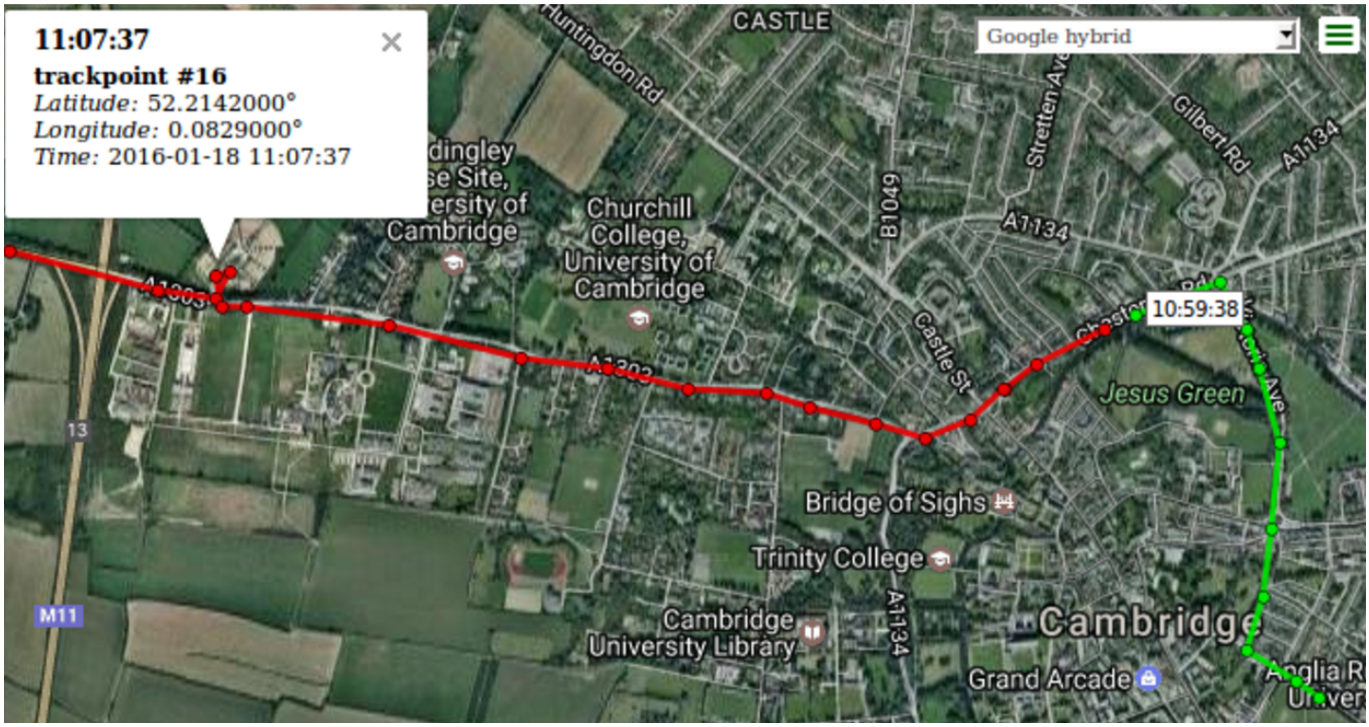
Often the timetable appears as a stand at the bus stop S_i. It shows multiple things, such as the bus arrival time at the same stop S_i, or a bus arrival time at the next stops $S_{i+1}, S_{i+2}, \ldots$ The way to present a timetable varies. For example, the information present on a stand is most likely displayed on some website.

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

1.3 GPS as a new Source

The current technology allows using 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 analyse 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 do the prediction analysis. I aim to build an algorithm which predicts the bus arrival times based on the most recent GPS data.

1.4 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.

Here, 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 this distance.

Chapter 2

Preparation

I see 3 main things that require thought before the implementation can commence:

1. What prediction algorithm I will code
2. What existing data and tools I will use
3. How will I evaluate the algorithm

Sections 2.1, 2.2, 2.3 explain each thing one by one.

2.1 Algorithm Choice

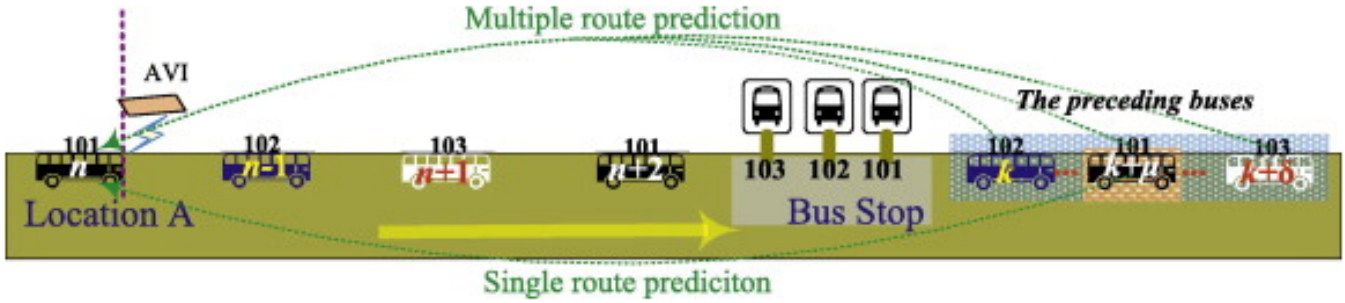
2.1.1 Existing Prediction Systems

The 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 predicting well for each of the different cases. For example, predicting the train arrival time is easy. Other vehicles do not block the train on the rail. The train speed is constant for most of the trip. Hence, the arrival time can be predicted dividing the remaining travel distance by its current speed. This approach works well for trains. However, I handle buses travelling either in one city or between multiple cities. The above approach is hopeless for the buses traffic because the current bus speed will quickly change due to the upcoming turns, traffic lights or other cars in front. Hence, just the momentary speed alone is of little use.

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

Firstly, I read a paper describing how other scientists are solving this problem:

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



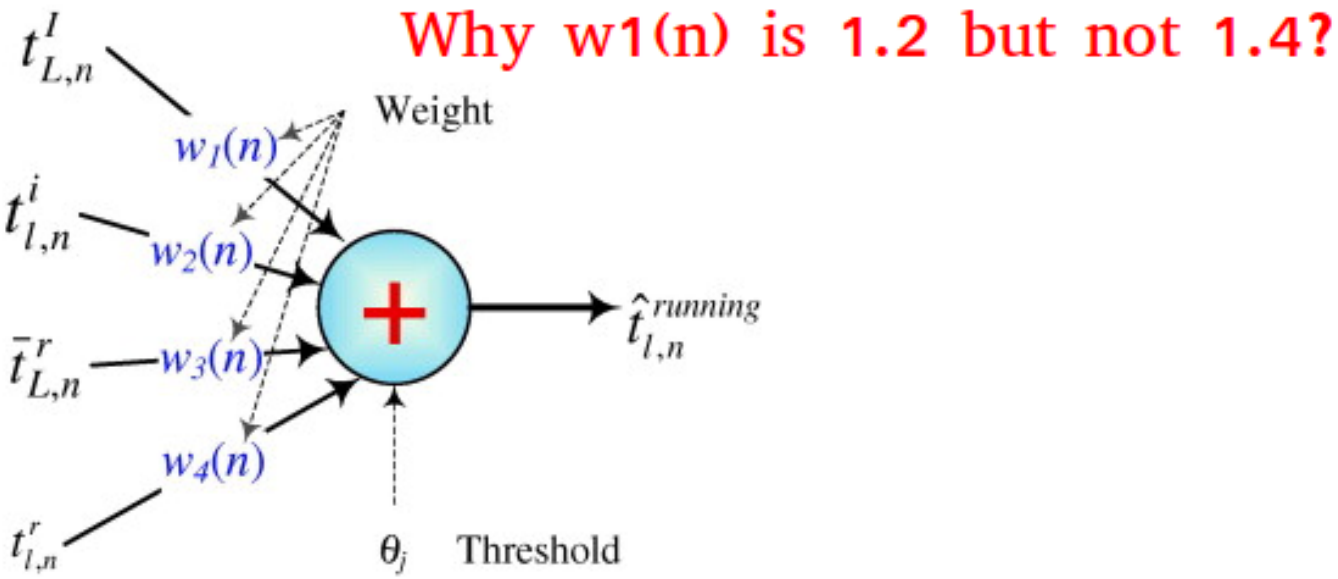
http://www.sciencedirect.com/science/article/pii/S0968090X11000155

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

My implementation incorporates one idea 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."*

2.1.2 My Choice

After talking with Prof Kelly, I decided not to use Machine Learning and found 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:



It is thus hard to explicitly re-program the model in case it does not work with the data. Since the traffic data generated by humans driving vehicles on a street can be very unpredictable (e.g. the Phantom Traffic Jam[1]), I see no guarantee that a particular model will succeed on the first attempt.

The model presented in this dissertation is much simpler to understand. I initially extract certain values (e.g. the time it takes for a bus to reach the stop B from the location A) from the historical

[1] https://www.youtube.com/watch?v=goVjVVaLe10

data set, just as the previous paper does. However, I do not feed the values into the ML model. I write the specific data-processing code instead. The result of this is that I have my own model[2] and understand it very well. If I see that the model does not predict well, I look at the data and figure out why that happens. I improve the model after inspecting the data.

2.2 Available Data and Tools

2.2.1 Starting Point

Initial GPS Data

A company named Vix is sending the real-time GPS data to the Cambridge servers. The GPS data shows how certain East-England buses (Stagecoach and Whippet) 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.

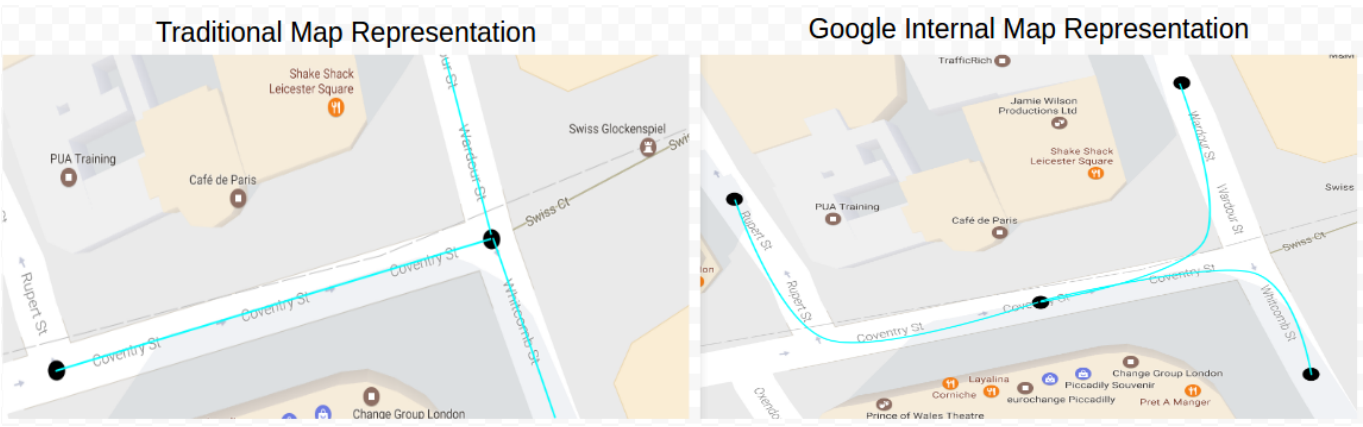
GPS Data for the Real-time System

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 that receives the new GPS data from the Vix company every 30sec.

The code that receives the real-time GPS data and immediately converts it to the JSON format was already written. I started this part by writing the new code to process the real-time JSON data.

2.2.2 Map Data

One can represent the road as an edge and the intersection of two roads as a vertex. Alternatively, Google Maps uses vertices to model roads and edges to model intersections joining two roads. They claim this unintuitive representation helps to reason that it takes a different amount of time to turn left, right or drive straight at the intersection:



[2]The model is presented in the implementation chapter.

In any case, a graph is used to represent the map. A bus trip corresponds to a walk along the graph edges. I initially considered predicting the arrival time by summing the time it takes for a bus to travel along every street on its route (i.e. along every edge on a path in a graph). This approach would have required me getting the map data. Collecting the data (street names, lengths, traffic directions, etc.) and building a graph representation of it is a big task. However, my overseers pointed out that it is **not necessary** to have this data at all:

*"My thought was more radical: ultimately what you need is to build a predictive model of when the bus will arrive based on previous samples of its position and other vehicles position. That model doesn't necessarily have to be based on the connectivity graph of the roads; it might be enough to (for example) pick out the mean speed of vehicles within any 200m square, and learn how bus arrival time (on any particular route) depends on all those speeds across the whole area. That wouldn't involve *extracting* map data at all."*

Following the given advice I decided not to collect the map data. This deviation from the original plan saved a lot of my time.

2.2.3 Programming Language Choice

A language must be convenient for the type of code I have to write. This project requires writing the following pieces of code:

1. Code inspecting the JSON format and converting it into another form.
2. Core algorithms (e.g. my route detection is a modification of the Dynamic Time Warping algorithm) predicting the arrival time.
3. Prediction **system design** code.

I think that Java is (up to a certain extent) a language convenient to write all of the above:

1. It has built-in *String* handling methods and IO facilities to read the text from a file.
2. It has advanced data structures (e.g. $HashMap$) to code an algorithm.
3. Java supports multithreading. Multiple components (the new GPS data processor, the arrival time predictor, the statistics generator) can run at the same time and interact with each other.

Thus, this project is coded in Java.

2.3 Evaluation Overview

I see two main evaluation metrics that are applicable for my project:

- Performance evaluation (i.e. how efficient my algorithm is).
- Precision evaluation (i.e. how accurate my algorithm is).

To evaluate the former, I am going to state the asymptotic complexity of the several system's components.

I concentrate on evaluating the latter in this project. I view the algorithm's precision as a metric to justify its **correctness** (i.e. if the algorithm claims that the bus will reach the bus stop at 3 pm, up to what extent this claim is correct). I follow the view that an algorithm has to be correct rather than fast in the first place. That is the reason why I concentrate on evaluating its precision.

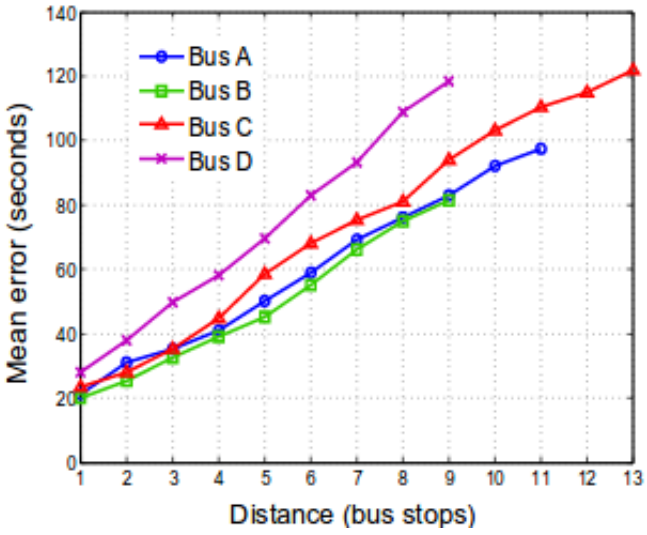
I read the second paper to gain more background how scientists evaluate prediction algorithms:

- Pengfei Zhou, Yuanqing Zheng, Mo Li: 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

I extracted the common evaluation metrics used in both papers. The Mean Absolute Error value (MAE) is the only sensible common evaluation metric. The MAE value is the basis of my evaluation chapter.

Also, the second paper illustrates one important aspect:



The prediction errors are meaningful only if one knows how far in advance they were generated. Suppose the prediction error is 80 sec. If it takes approximately one hour to arrive at the last bus stop, then the 80 sec. prediction error for the last stop is a small number. If I predict the arrival time at the next stop less than 2 minutes away from the current location, then the 80 sec. prediction error is a large number.

My evaluation chapter takes this aspect into account.

Chapter 3

Implementation

3.1 Data Preprocessing

Below is the starting point input data example:


```
{"vehicle_id":"175","latitude":51.88656,"longitude":-0.43085587,"timestamp":1478775682}
{"vehicle_id":"187","latitude":51.54462,"longitude":-0.17597081,"timestamp":1478775790}
{"vehicle_id":"188","latitude":51.498653,"longitude":-0.14752254,"timestamp":1478775790}
{"vehicle_id":"189","latitude":51.532436,"longitude":-0.17194784,"timestamp":1478775789}
{"vehicle_id":"191","latitude":51.882435,"longitude":-0.42654553,"timestamp":1478775750}
{"vehicle_id":"193","latitude":51.88799,"longitude":-0.45470643,"timestamp":1478775812}
{"vehicle_id":"196","latitude":51.889244,"longitude":-0.5213731,"timestamp":1478775810}
```


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


```
{"vehicle_id":"175","latitude":51.88656,"longitude":-0.43085587,"timestamp":1478775682}
```


means that a vehicle nr. 509 was at the geographical position $(51.88656, -0.43085587)$ on the 10st November 2016, GMT time 11:01:22 (the time is calculated from the timestamp field 1478775682).

A new snapshot file arrives once in 30s. The data preprocessing part takes multiple input files (e.g. I preprocess $3600 \cdot 24/30 = 2880$ files received in **one** day to build historical trips for that day) and converts them into another format:

day10_bus18_subtrip4

```
time,latitude,longitude
2016-11-10 12:45:34,52.22437,-0.28086
2016-11-10 12:46:28,52.22697,-0.27439
2016-11-10 12:47:03,52.22769,-0.27152
2016-11-10 12:47:33,52.22778,-0.27109
2016-11-10 12:53:25,52.22813,-0.25930
2016-11-10 12:53:25,52.22813,-0.25930
2016-11-10 12:54:23,52.22822,-0.25471
2016-11-10 12:54:23,52.22822,-0.25471
2016-11-10 12:56:03,52.23279,-0.25212
2016-11-10 12:56:33,52.23450,-0.24925
2016-11-10 12:56:33,52.23450,-0.24925
2016-11-10 12:57:34,52.23441,-0.24609
2016-11-10 12:58:04,52.23154,-0.24838
2016-11-10 12:59:03,52.23091,-0.24867
2016-11-10 12:59:33,52.22796,-0.25011
2016-11-10 13:00:03,52.22724,-0.25169
2016-11-10 13:00:33,52.22634,-0.25212
2016-11-10 13:01:03,52.22356,-0.25485
...
```

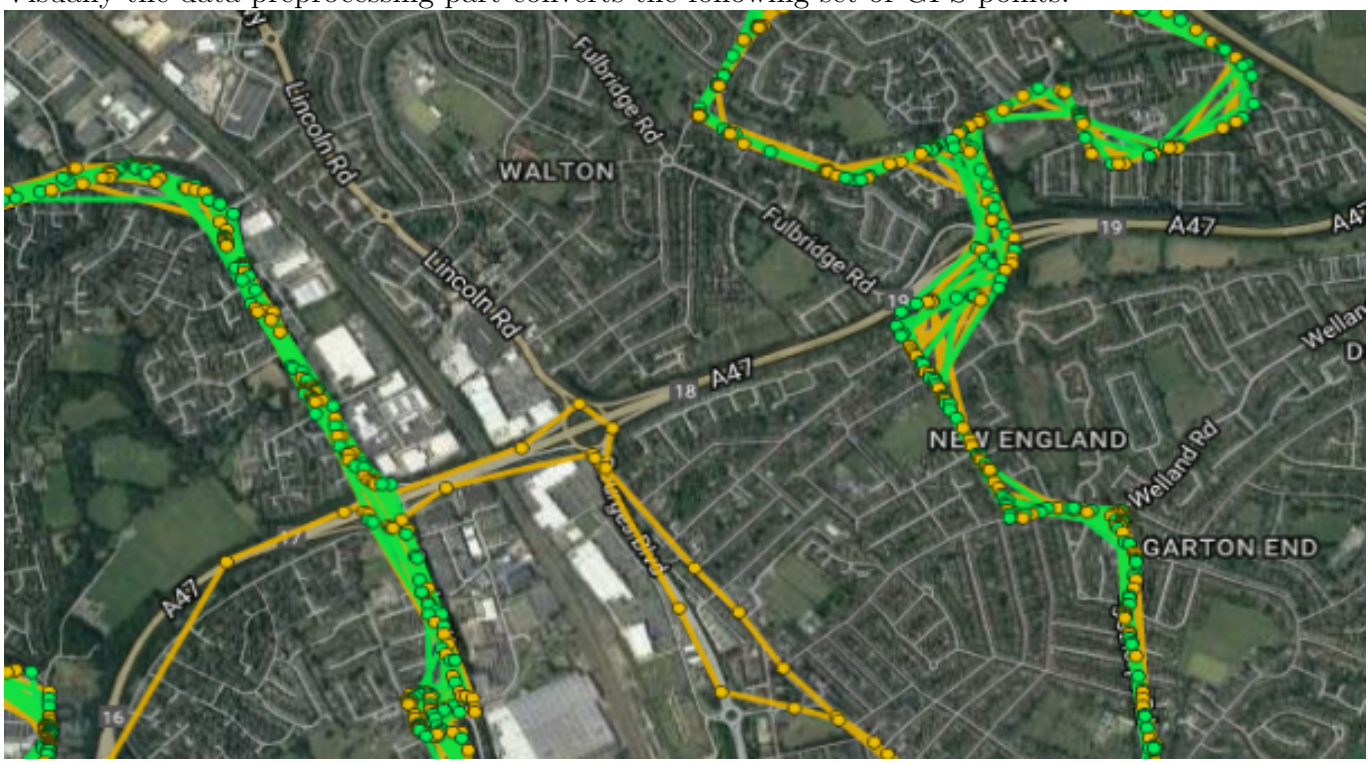
day10_bus18_subtrip5

```
time,latitude,longitude
2016-11-10 15:48:04,52.32939,-0.20025
2016-11-10 15:48:35,52.32930,-0.19968
2016-11-10 15:49:05,52.32769,-0.19853
2016-11-10 15:49:34,52.32679,-0.19997
2016-11-10 15:50:04,52.32446,-0.20399
2016-11-10 15:50:35,52.32294,-0.21175
2016-11-10 15:51:04,52.32303,-0.21749
2016-11-10 15:51:34,52.32410,-0.22137
2016-11-10 15:52:04,52.32536,-0.22655
2016-11-10 15:52:36,52.32670,-0.23014
2016-11-10 15:53:05,52.32831,-0.23287
2016-11-10 15:53:36,52.32966,-0.23991
2016-11-10 15:54:06,52.33073,-0.24264
2016-11-10 15:54:36,52.33190,-0.24522
2016-11-10 15:55:06,52.33234,-0.24709
2016-11-10 15:55:35,52.33252,-0.24982
2016-11-10 15:56:05,52.33270,-0.25557
2016-11-10 15:56:34,52.33217,-0.26419
...
```

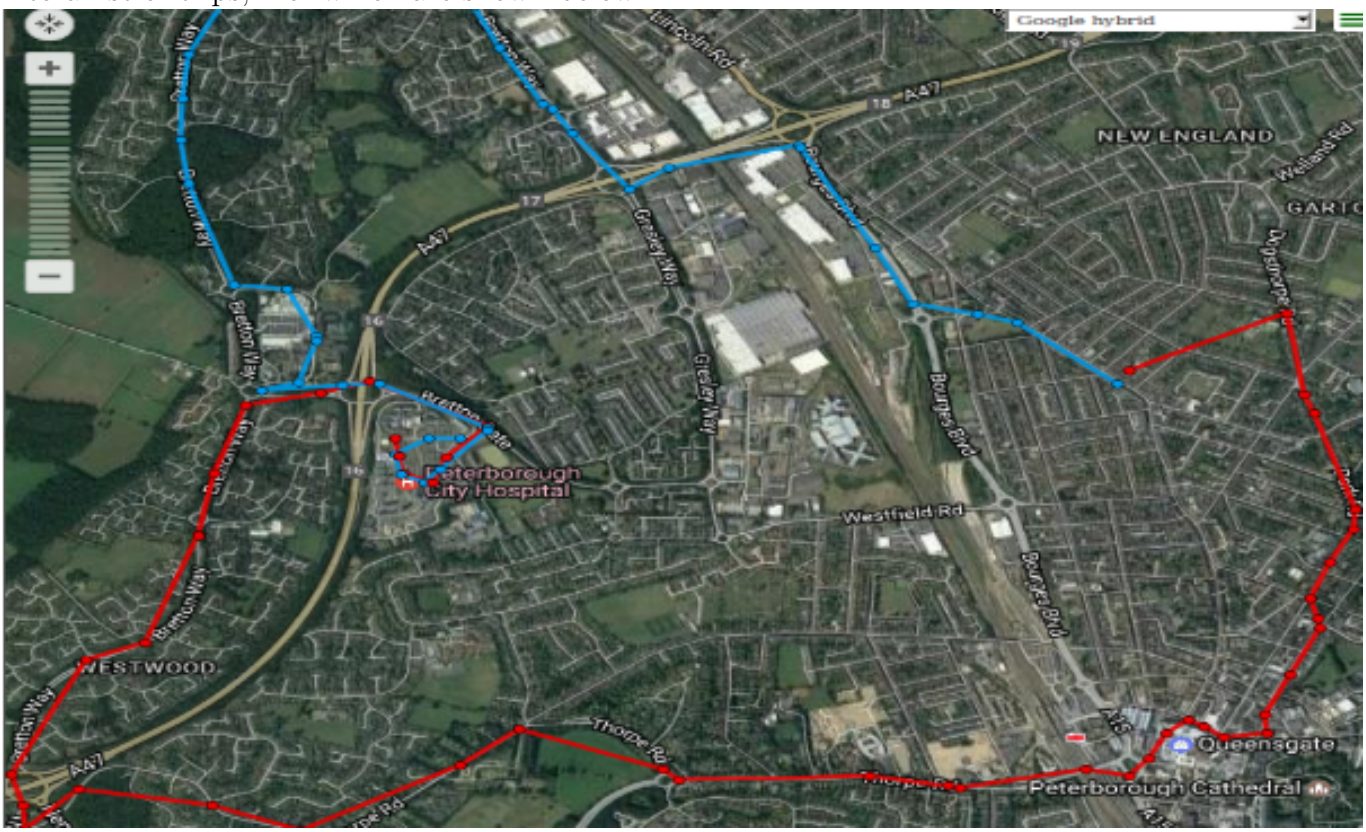
At the end, I have a file for each trip made by one bus in one day[1].

[1] 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:



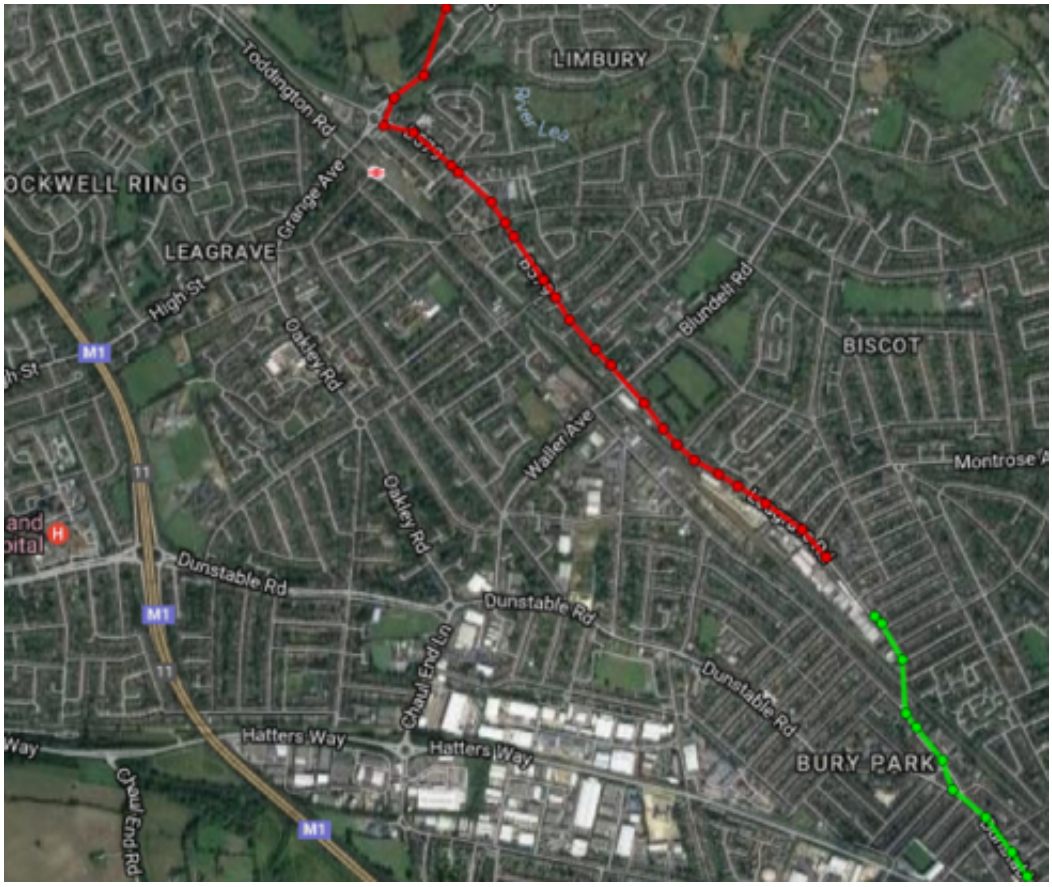
into a list of trips, 2 of which are shown below:



The preprocessing part helps to inspect the data. I can look at a separate trip (or merge multiple trips of my choice) and understand why the buses are moving in this way. The trips I obtain are used in further implementation sections.

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** the bus will travel next. It turns out that in my case getting this information is not straightforward:

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":"37","label":"CBL-301","latitude":51.878403,"longitude":-0.41260874,"timestamp":1478775871}

An extra field **label** names the route a bus follows. Auxiliary data might indicate what route the label **CBL-301** 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. The GPS device on the actual vehicle is responsible only for providing the latitude and longitude coordinates. The device is not responsible for providing 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 (a bus is unlikely to follow the same route every day) and the data does not always indicate that.

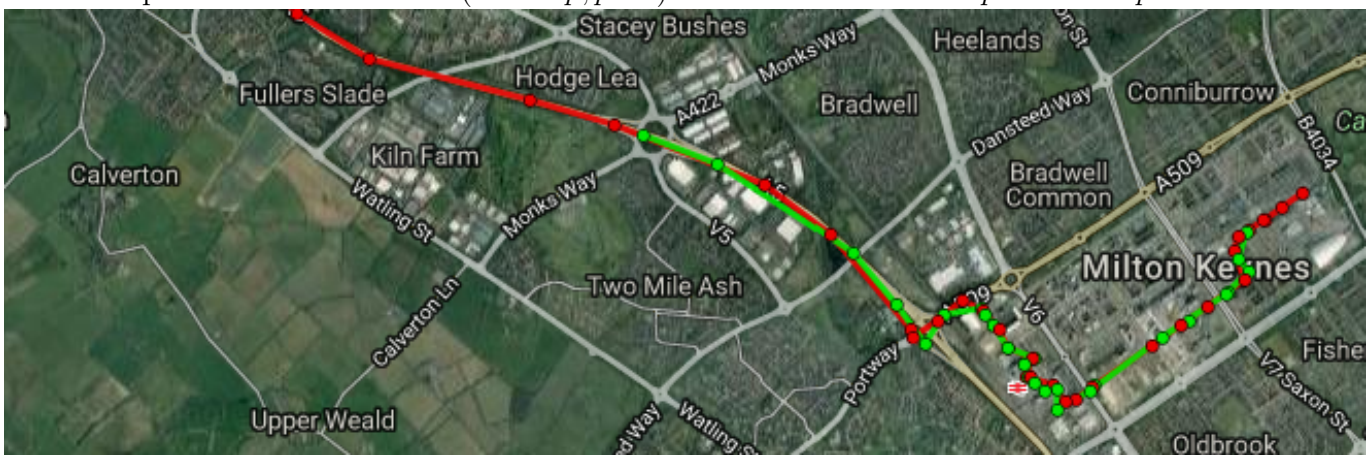
- The static data only shows the sequence of bus stops. It does not show a detailed sequence of streets a bus will follow. It is much better to know an actual path for the accurate arrival time prediction.

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 (from now on referred to as a boolean predicate) that can tell whether one sequence of GPS points follows another.

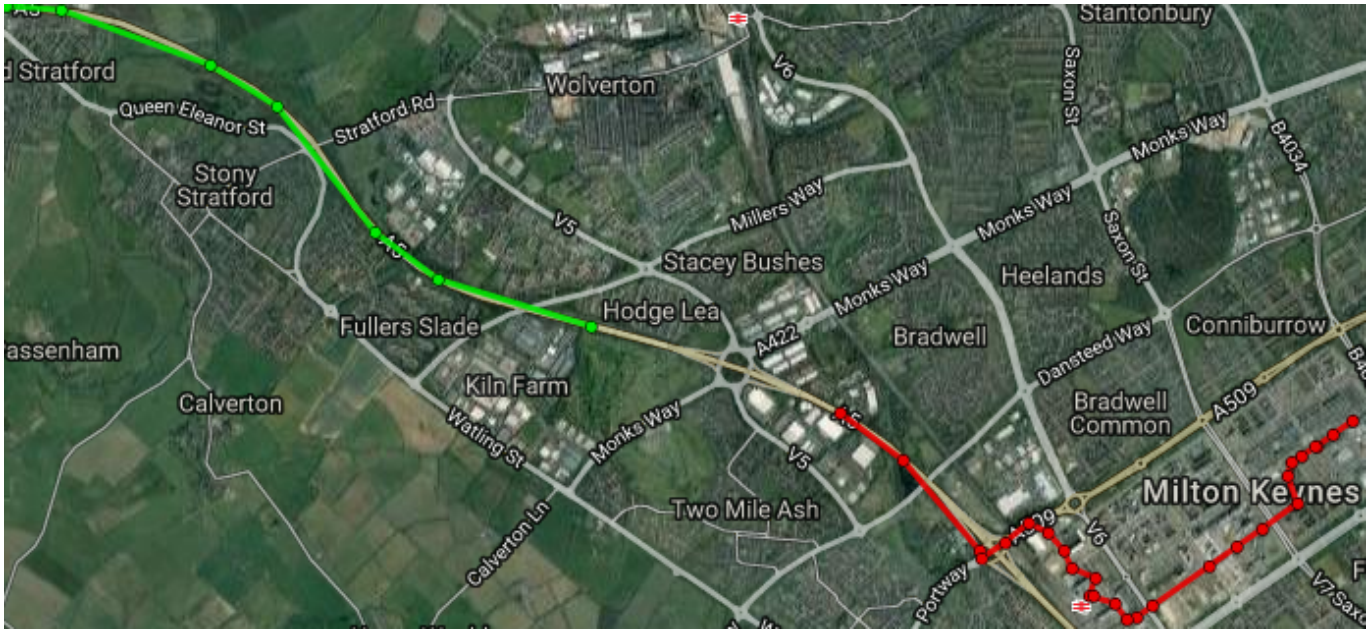
3.2.2 *FollowsPath* Predicate Definition

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



The $FollowsPath(subtrip, path)$ predicate returns true for the picture above, because it is intuitively clear that the subtrip follows a path.

The $FollowsPath(subtrip, path)$ predicate returns false for the picture below, as the two GPS sequences do not intersect at all:



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 function (taking a point and a segment as an input) defined as

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

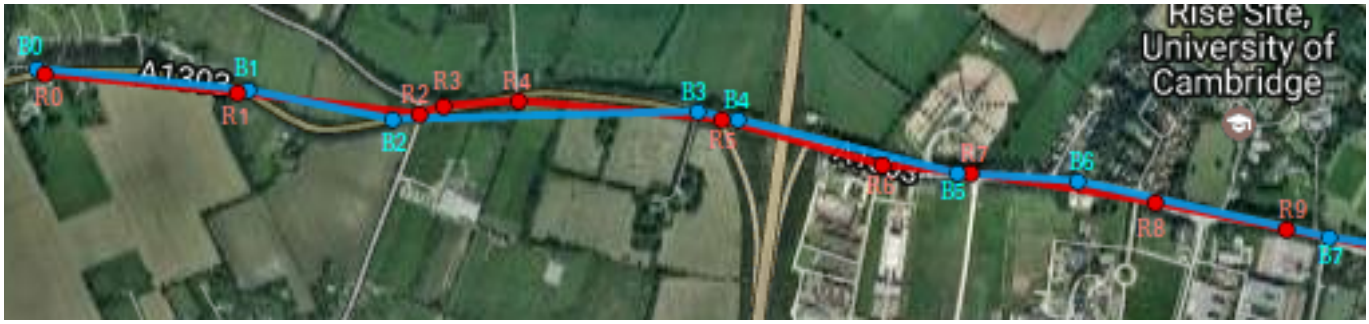
equals 0. For each point P_i, I find a segment S_{P_i} such that $err(P_i, S_{P_i})$ is minimised. The *FollowsPath* predicate returns true iff

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

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. The *FollowsPath* predicate returns true in this case. That 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:

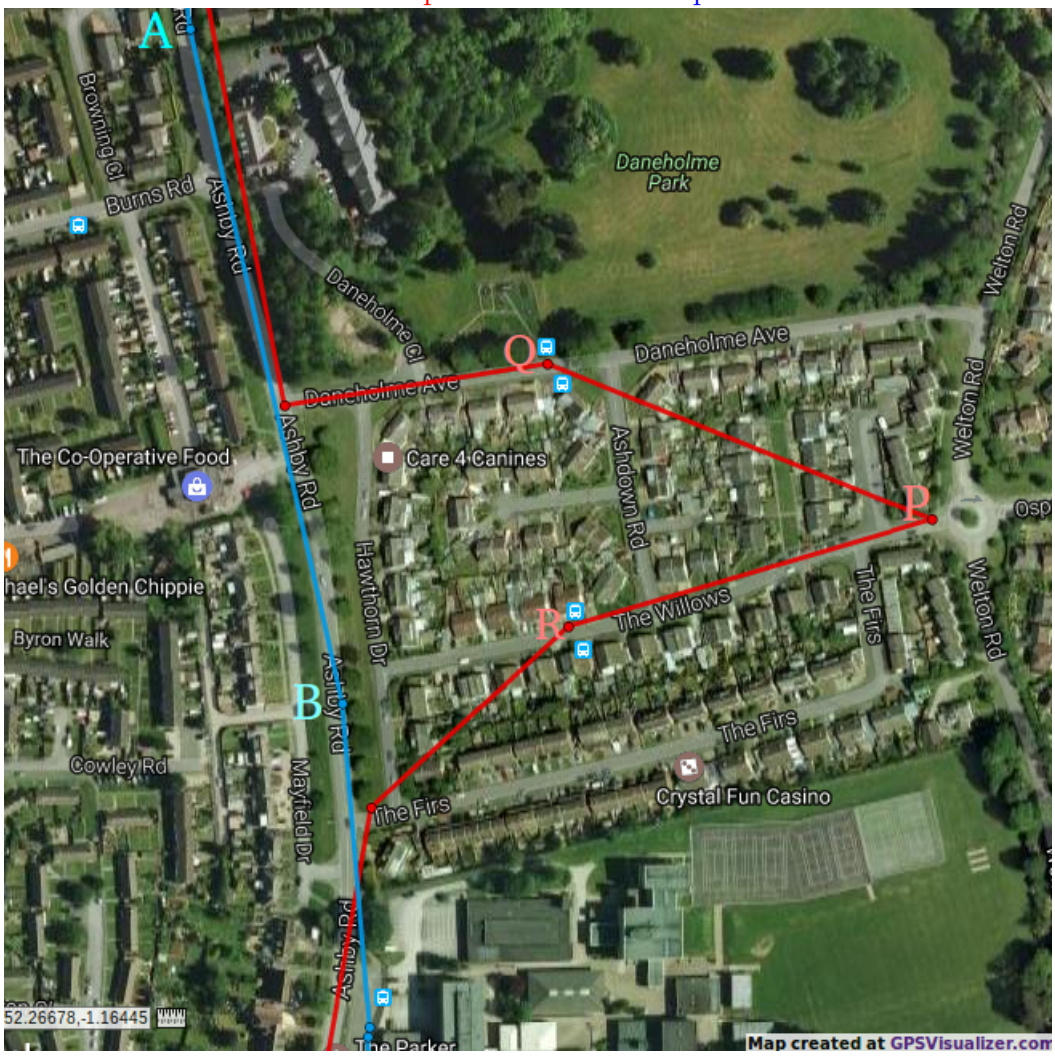


In this example I am aligning a sub-trip (R_0, ..., R_9) with a path (B_0, ..., B_7, ...). The computation of S proceeds as follows:

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

and the predicate again returns true, as desired.

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



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

$$err(P, AB) = (12.9 + 9)/10^2 - 1 = 1.1$$

This error is more than 10 times larger than the typical values less than 0.1. Furthermore, the predicate generates large errors for each sub-trip's point that deviated from the path (e.g. for the points Q and R in the diagram). Hence, the total errors sum S exceeds $n\epsilon$ and the predicate returns false.

I present the practical evidence that the *FollowsPath* predicate works in the evaluation section.

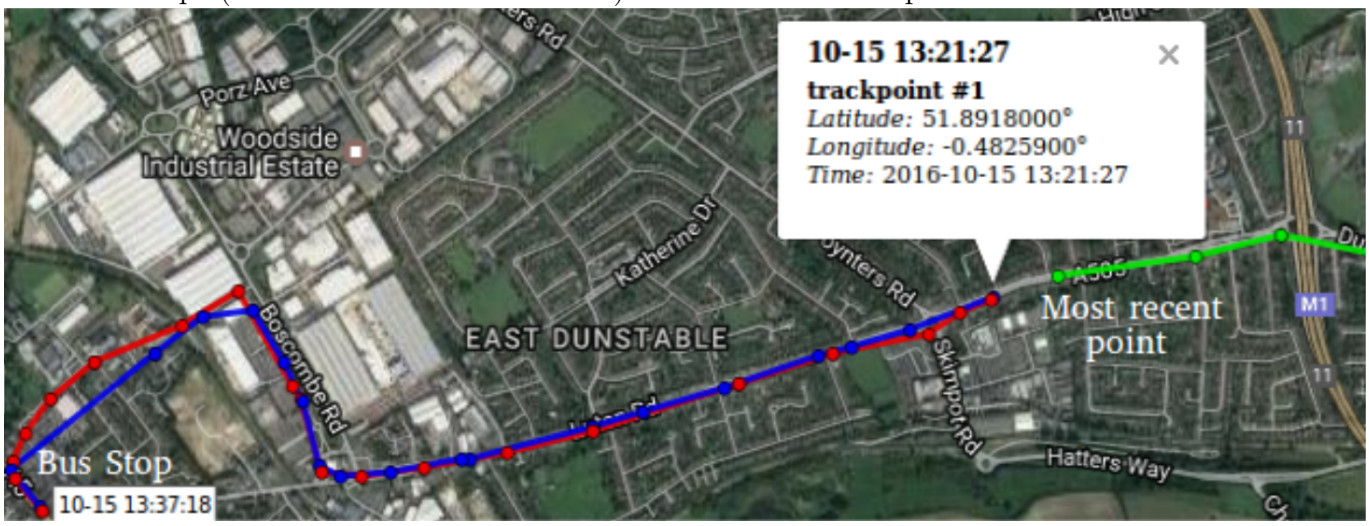
[2]The distances 12.9, 9 and 10 are scaled here. Scaling does not change the ratio value.

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:



I show 2 example historical sub-trips (blue and red) in the picture above. Both of them start at the current vehicle's location (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. I return the median duration value 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 the 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. The outlier can negatively influence the result if I compute it differently.

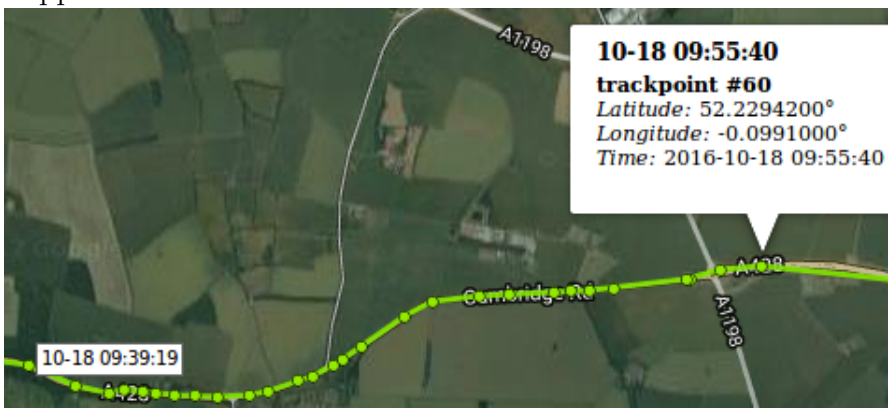
The median value predicted better compared to the arithmetic mean during the testing phase. Thus, I decided to stick with the median value.

3.3.2 Optimisation Nr. 1

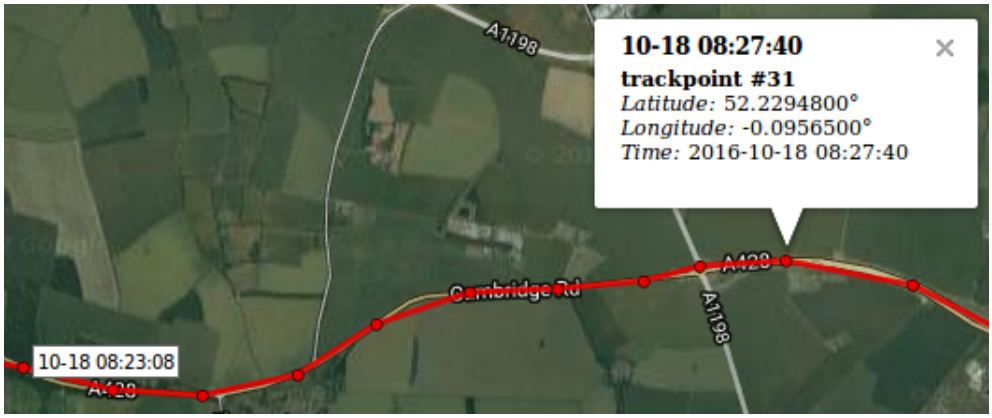
If in the historical trips set certain trips have **just** travelled along the same route (e.g. in the last 30min.), I predict the median time just from those recent trips. Otherwise, I use the basic approach.

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 completely different amount of time (e.g. 4 min.). This can happen due to a different traffic congestion level in a different time of the day. Predicting the arrival time using these historical trips can be misleading. This optimisation discards the 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 it reached the current bus location. If P is roughly the same location as the bus location of the current bus M minutes 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 apply the basic approach on it and generate to predict the arrival time.

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 yesterday's trips 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). A week period does not fix the problem either. 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.

Fortunately, my optimisation addresses both the fact that the traffic follows some periodic pattern and 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 correlates with the traffic congestion. The reason why some historical trip (that 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 traffic periodicity, then this optimisation detects all periodic trips, regardless of the actual period value.

Optimisations are tested in the evaluation section.

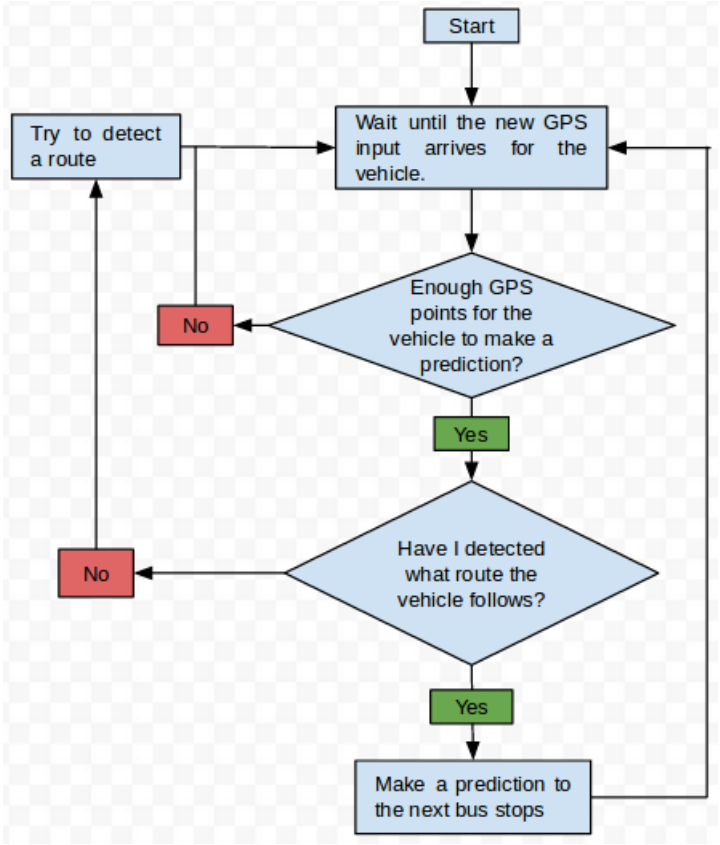
3.4 Real-time System (Optional)

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[3] 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.[4]

(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).

[3]Such as another student next year.

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

Chapter 4

Evaluation

4.1 Evaluation Data Preparation

I take the historical November data and generate the set S consisting of 532428[1] trips. I also take 3050 routes (2640 of them are read from old files, 410 of them are new routes present in the new Traveline National Dataset (TNDS)).

I then run a long computation (it took 36 hours to complete on Dr Lewis machine):

For each trip $t \in S$, I search for a route that the trip **completely** went through (i.e. visited all route stops in order). If I find a route, I save the trip. If no route is found, I discard the trip. At the end, I save only the routes for which more than 100 trips are found.

I am very selective during this computation. I consider that the bus visited the stop s iff the trip contains a GPS point very close to s.

The computation returns 25 routes each having more than 100 trips passing through it completely. The next sections evaluate my algorithm on these 25 routes.

It is surprising that **no** route is from the new TNDS set. I contacted the officer working with this data and got the following reply:
"In practice, within the TNDS, you will find that Route data is either limited or non-existent."

The process above voluntarily discards a lot of data. I want to explain this decision. Improving any system is a natural ambition. I see two ways how the prediction algorithm can perform better:

- The algorithm itself is improved.
- The data becomes better.

If **both** options are possible, I think one should **first** try the latter. This case applies to my project. I want to evaluate the algorithm on data that meets the quality standards. Thus, I discard the poor data and use only the qualified data for the evaluation.

[1]The size of this set is large but sensible. If 1000 buses are present in each JSON file, the duration of 1 full trip is $\sim 1h$, and November has 30 days, then $30 \cdot 24 \cdot 1000 = 720000$ is comparable to 532428.

4.2 Route Detection Evaluation

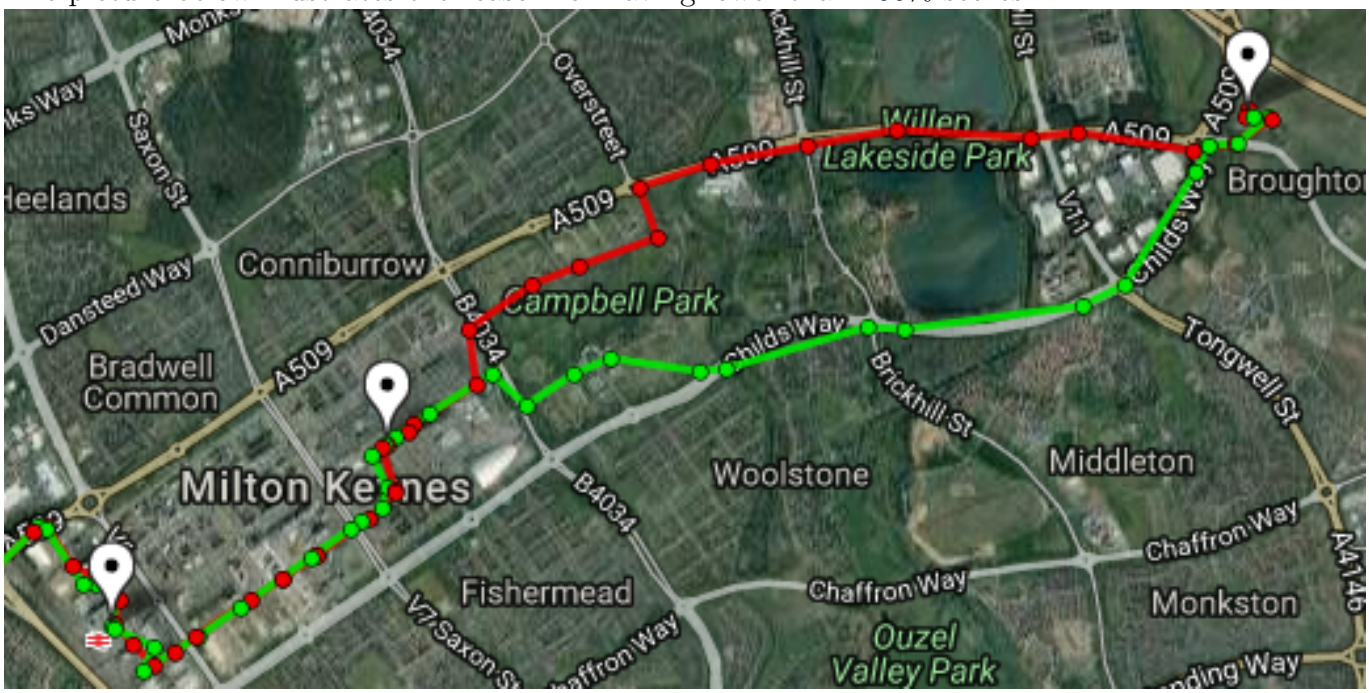
4.2.1 Sensitivity Score

This section checks whether the $FollowsPath(sub\text{-}trip,\ path)$ predicate returns *true* when the *sub-trip* indeed follows a *path*. Each path p (that corresponds to a route) has a set T_p of trips that went through it. I take a sub-trip s of each trip $t \in T_p$ and check that the $FollowsPath(s, p)$ returns true.

For 11 routes (out of 25) the $FollowsPath$ predicate has 100% success rate.

I present the sensitivity scores for other 14 routes in the decreasing order:
(0.99, 0.99, 0.98, 0.98, 0.89, 0.86, 0.86, 0.84, 0.58, 0.48, 0.38, 0.13, 0.12)

The picture below illustrates the reason for having lower than 100% scores:

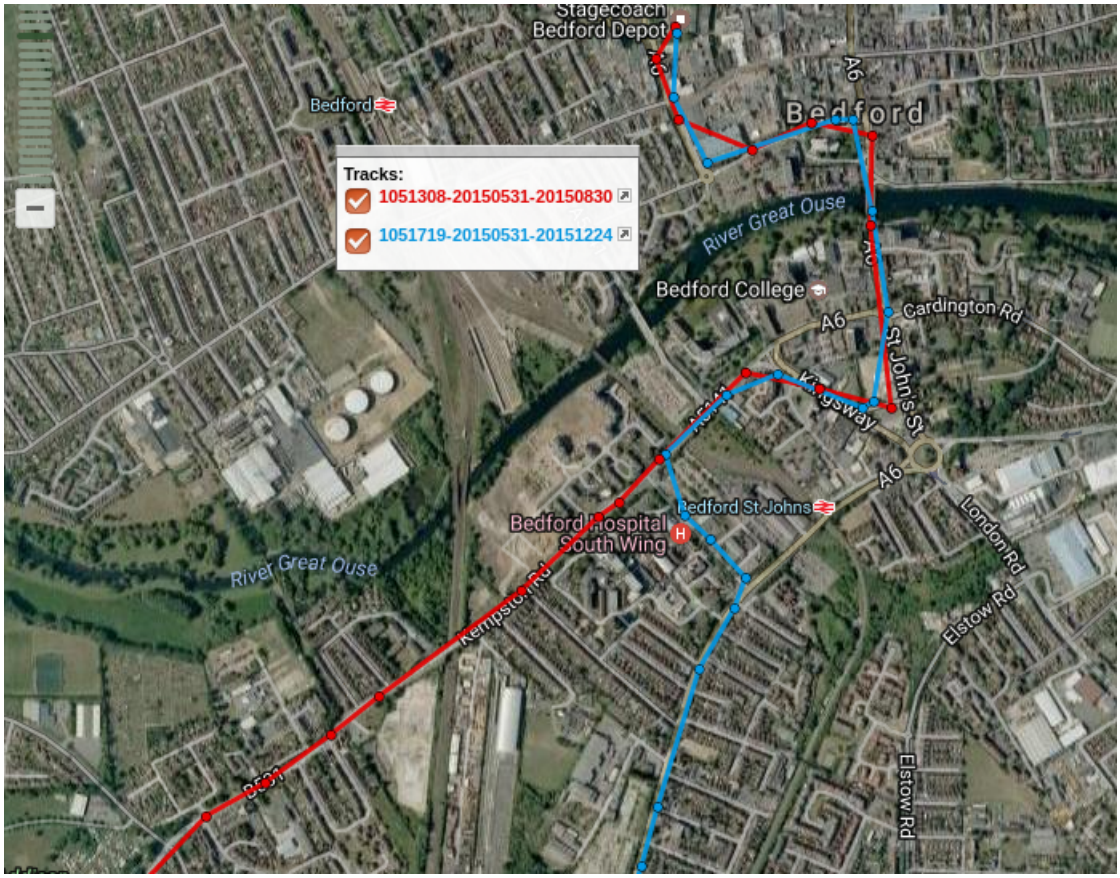


The trip reached the second bus stop using a different path than the one I have as a path representing the route. Some routes have few stops (e.g. 3) far away from each other. There are many different paths a bus can use to travel between two consecutive stops in this case. That is the reason why for some routes the sensitivity score is low.

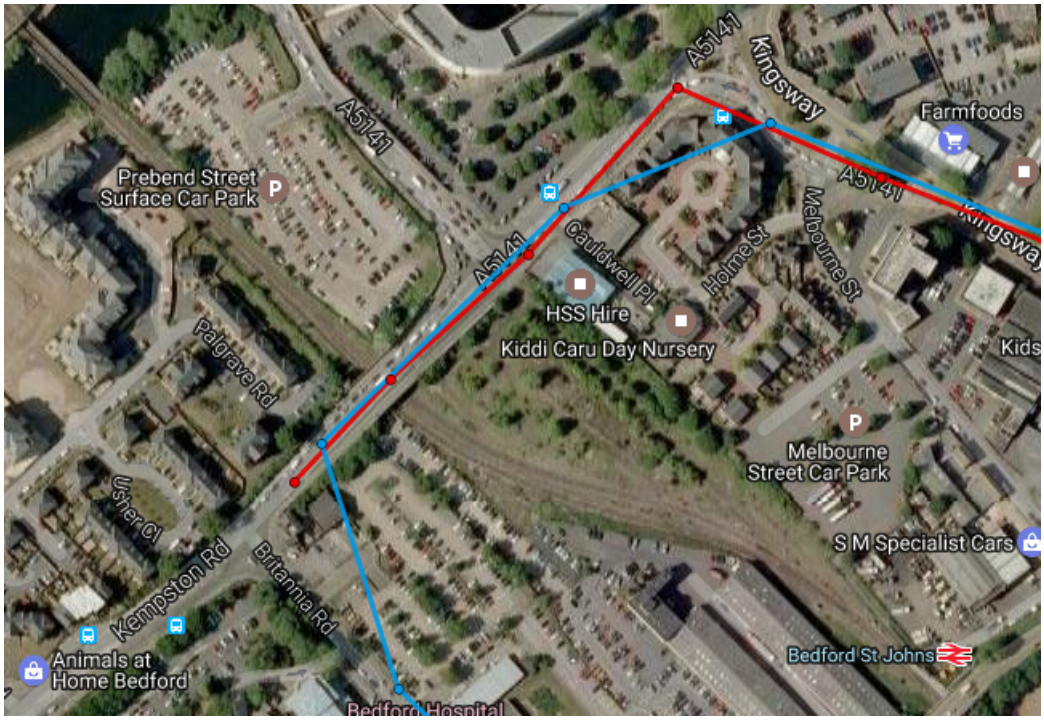
4.2.2 Specificity Score

This section checks 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. 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 *sub-trip* of the first 12 points (each trip deviates from a path approximatelly after 12 GPS points) and check whether the $FollowsPath$(*sub-trip*, *path*) returns false. Out of 107 trips only once the predicate incorrectly returns true. The misclassified *sub-trip* has not deviated from a *path* yet for a predicate to detect that:

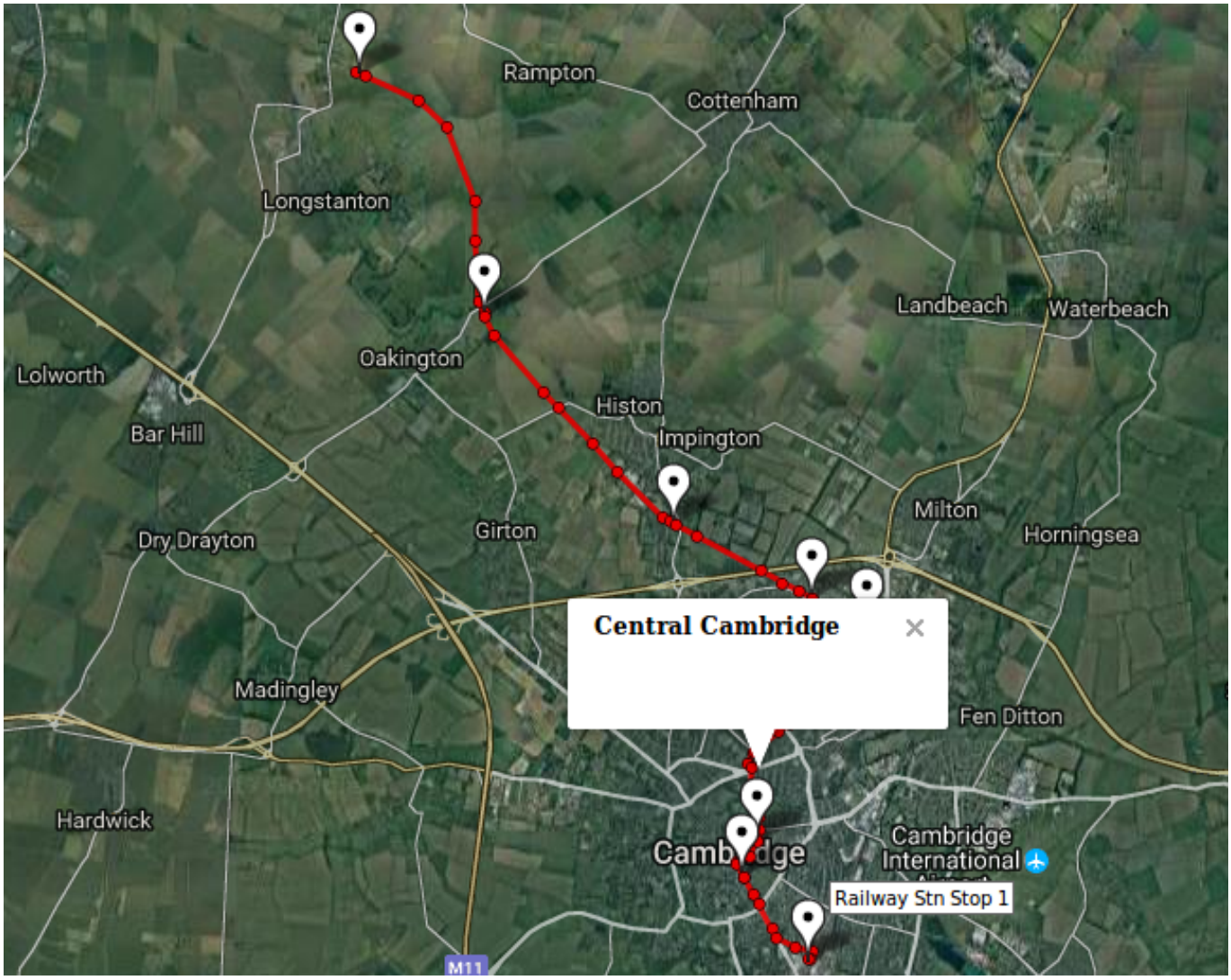


The high sensitivity and specificity scores show that the $FollowsPath$ predicate works **if** used in the right place at the right time.

4.3 Arrival Time Prediction Evaluation

4.3.1 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. I will use the example route 1046531-20150531-20150830 for the presentation. The route begins outside of Cambridge and ends at the Cambridge Railway station:



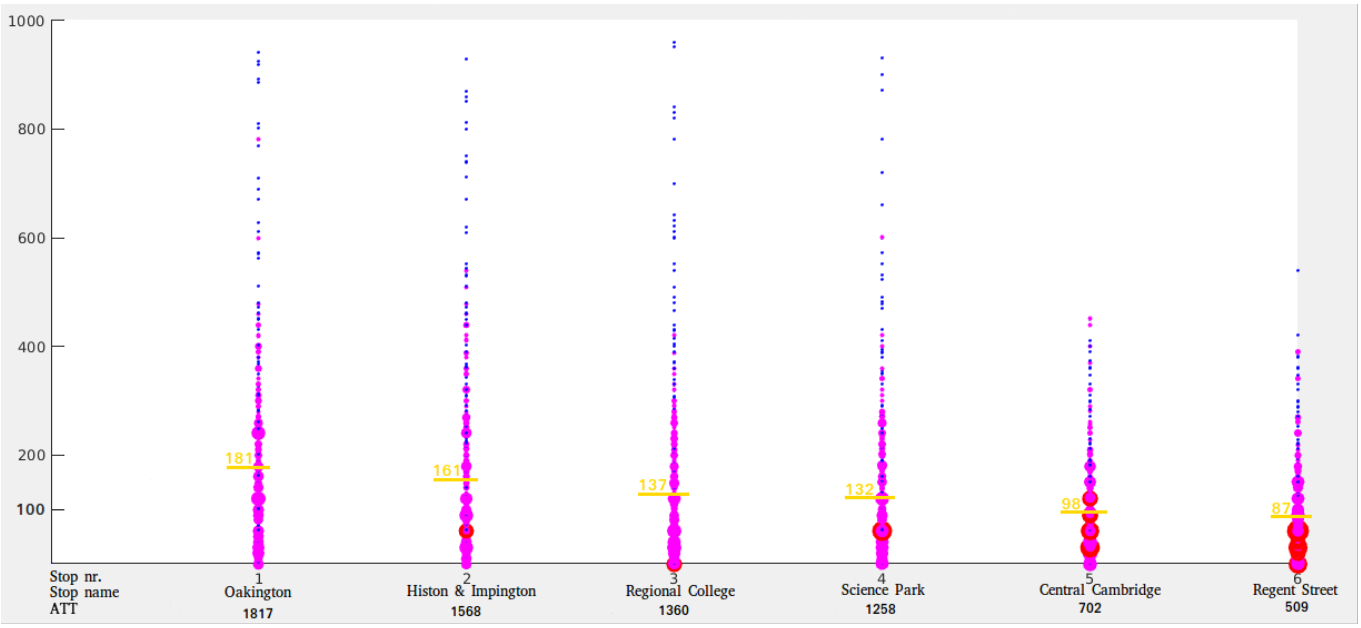
(i) **The MAE value**. Suppose I have a set of trips T following a particular route. Each trip eventually reaches a bus stop of interest. Suppose for trip $t \in T$ the algorithm predicts the bus arrival time at stop to be $t_{predicted}$, but the bus actually arrives at t_{actual}. The absolute prediction 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 674 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. I then read the rest of the history and check when

the trip has actually arrived at the Railway Station. The resulting MAE value is **98** sec. for this example route. 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 $MAE_A < MAE_B$. I will evaluate the optimisations based on this claim.

(ii) **The scatter plot**. The MAE value gives the 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:

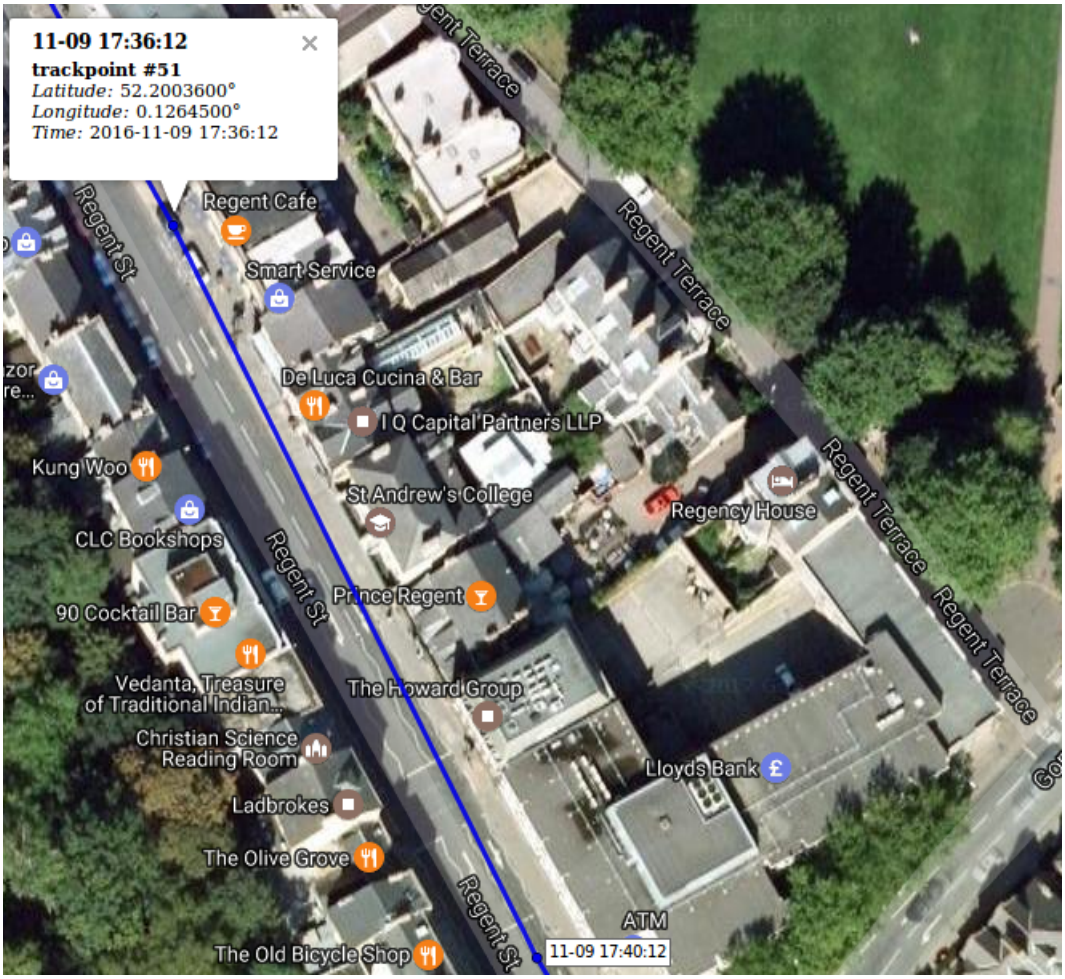


The x-axis labels the stop number i. I predict when the bus will reach the Railway Station (the last stop) starting from the i-th stop. The y-axis labels the absolute prediction error (measured in seconds). A point (i, y) tells that for some trip the prediction from the i-th stop to the last stop has an abolute error equal y. I plot a larger point whenever I have more repeated errors (i, y). I also show the MAE value (labelled in yellow) and the average travel time[2] from the i-th stop to the last stop in the diagram for completeness.

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.
- Many prediction errors are smaller than the MAE value in my case. There are a few large errors that make the MAE value increase. The large errors occur due to unusual traffic patterns:

[2]For each trip I calculate the time it actually takes to reach the last stop. I then return the mean value.



For some bus it took 4 minutes to pass just a few buildings. Most historical trips usually pass this segment in a much shorter time (e.g. 30s). Hence, 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. It does not show 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 is constructed. Half of the errors are positive, half negative on average. 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.

I evaluate the prediction algorithm for each of the time intervals separately. I predict the arrival time to the last stop starting from each of the other stops. The **table of errors** shows the aggregated results:

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

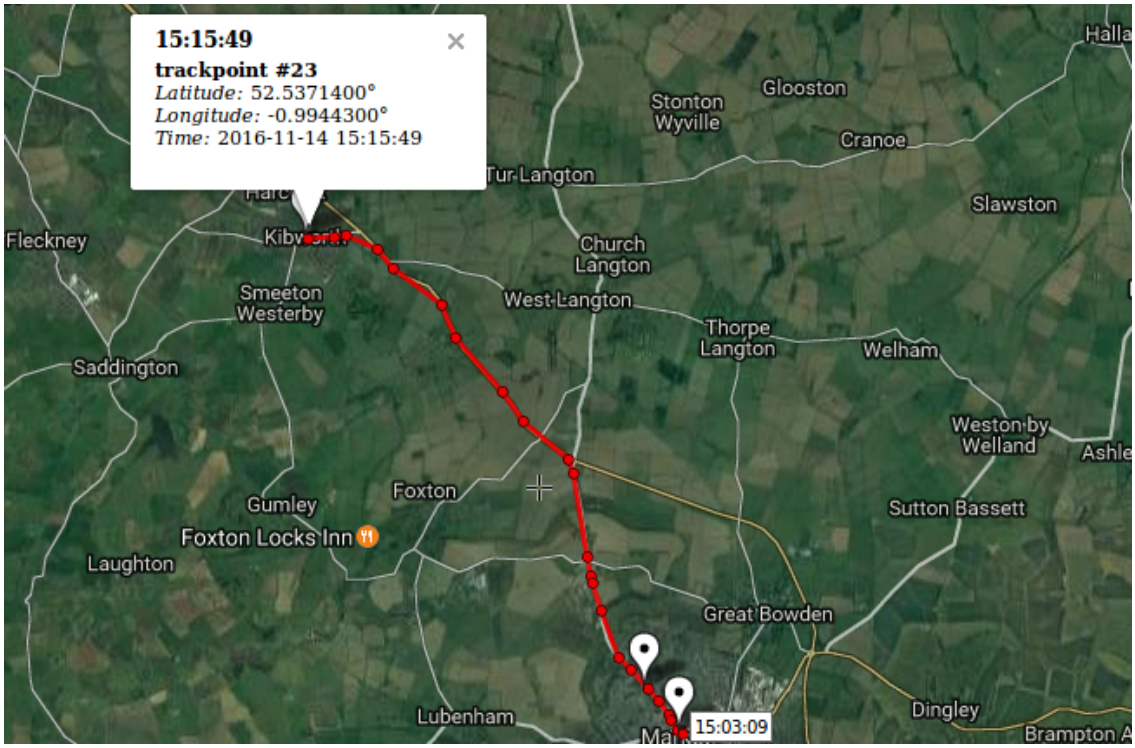
4.3.2 Multiple Routes Evaluation

I used one route to present the metrics. The fact that an algorithm works on one route does not neccessary imply that it works on others. Hence, I evaluate the algorithm on all other routes for which I have enough data (25 in total). The detailed results are given in the appendix B.1.

Good Route Example

Route 1056213-20150614-20151223		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	77	61	70	83
The Square	ATT	782	720	735	744
From stop nr. 2	MAE	85	46	37	49
Police Station	ATT	642	595	608	618
Last stop: The Square		7 trips used	62 trips used	31 trips used	48 trips used

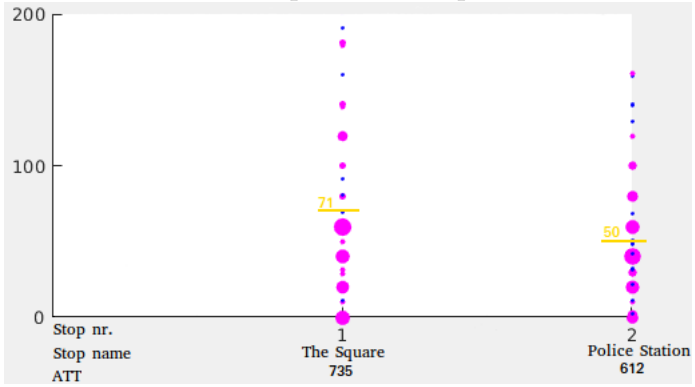
The algorithm performs well on the route 1056213-20150614-20152323:



The algorithm predicts the arrival time with an average error less than 10% for many time intervals. I believe that a long highway path is the reason, why the algorithm is successful for this route. Firstly, highways have a feature that they they are less connected with other roads. 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. Other roads do not affect the highway traffic as much as they affect the city traffic. Hence, only historical trips of the buses travelling along this route are enough to generate reasonable predictions (note that my prediction algorithm uses only historical trips following this route).

Secondly, the bus speed on the highway does not vary much. Suppose that for the last 5 minutes the bus speed was 60mph. It is likely that the bus will keep travelling at roughly the same speed. Contrast this with the case when the bus is on a street in a city. The bus speed will fluctuate a lot due to traffic lights, other cars causing a congestion, pedestrians crossing the street. 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[3] 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.

I include the scatter plot for completeness:



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

Bad Route Example

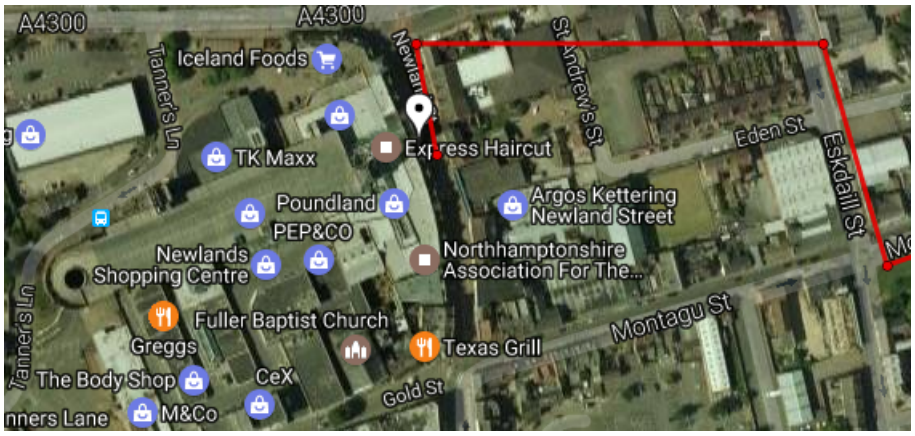
The algorithm performs poorly on the route 999024-20150526-20150830:

Route 999024-20150526-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	135	121	141	136
Newland Centre	ATT	640	664	632	603
From stop nr. 2	MAE	45	39	43	38
Bonham Court	ATT	292	305	271	268
From stop nr. 3	MAE	36	34	26	29
Havelock Street	ATT	217	231	202	205
From stop nr. 4	MAE	32	26	18	37
Lobelia Road	ATT	166	186	164	158
From stop nr. 5	MAE	20	21	21	35
Walnut Crescent	ATT	101	115	103	115
Last stop: Aster Road		25 trips used	166 trips used	23 trips used	10 trips used

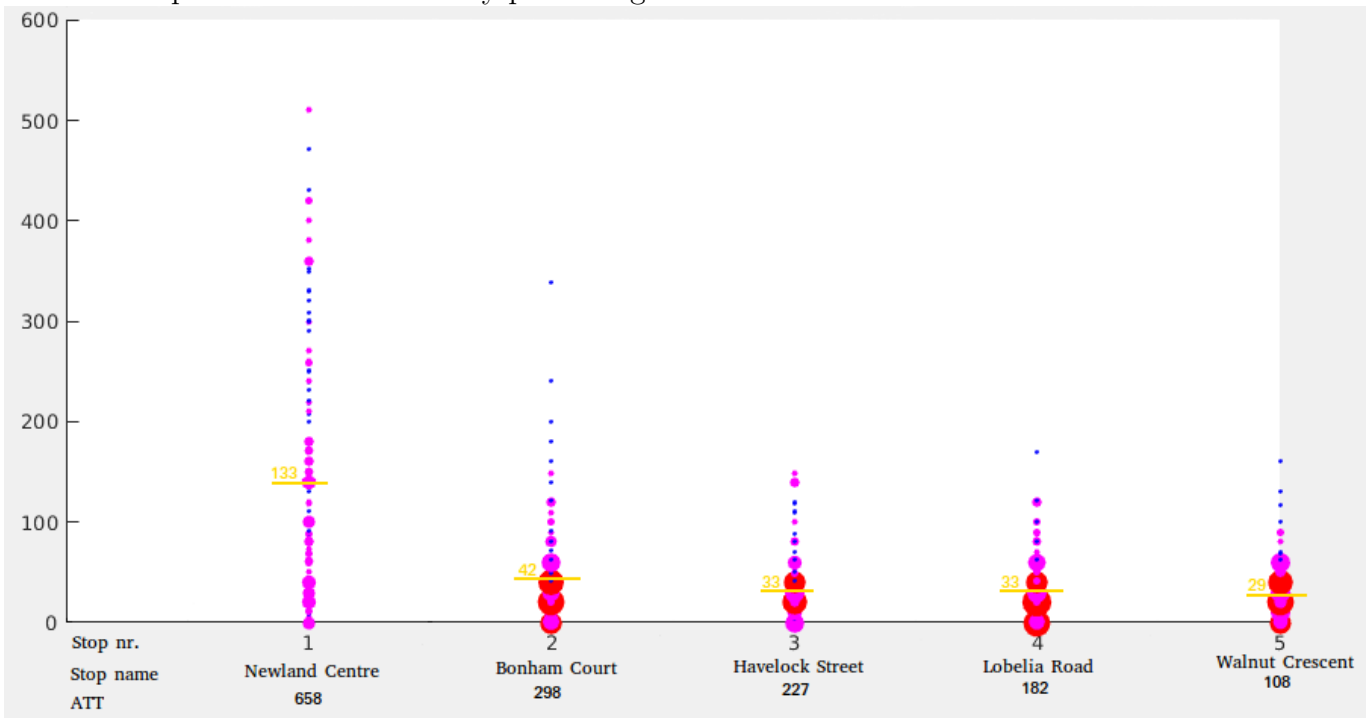
The poor performance is due to the following reasons. Firstly, there are many bus stops a bus frequently[4] stops at. The amount of time a bus is standing at one stop affects its arrival time at the future stop. Unfortunately, the algorithm does not explicitly deal with the bus standing time.

[4]Every 2min. or so.

Secondly, the route includes the crowded city places (e.g. 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. The errors spread out across the whole range of y values instead. 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)

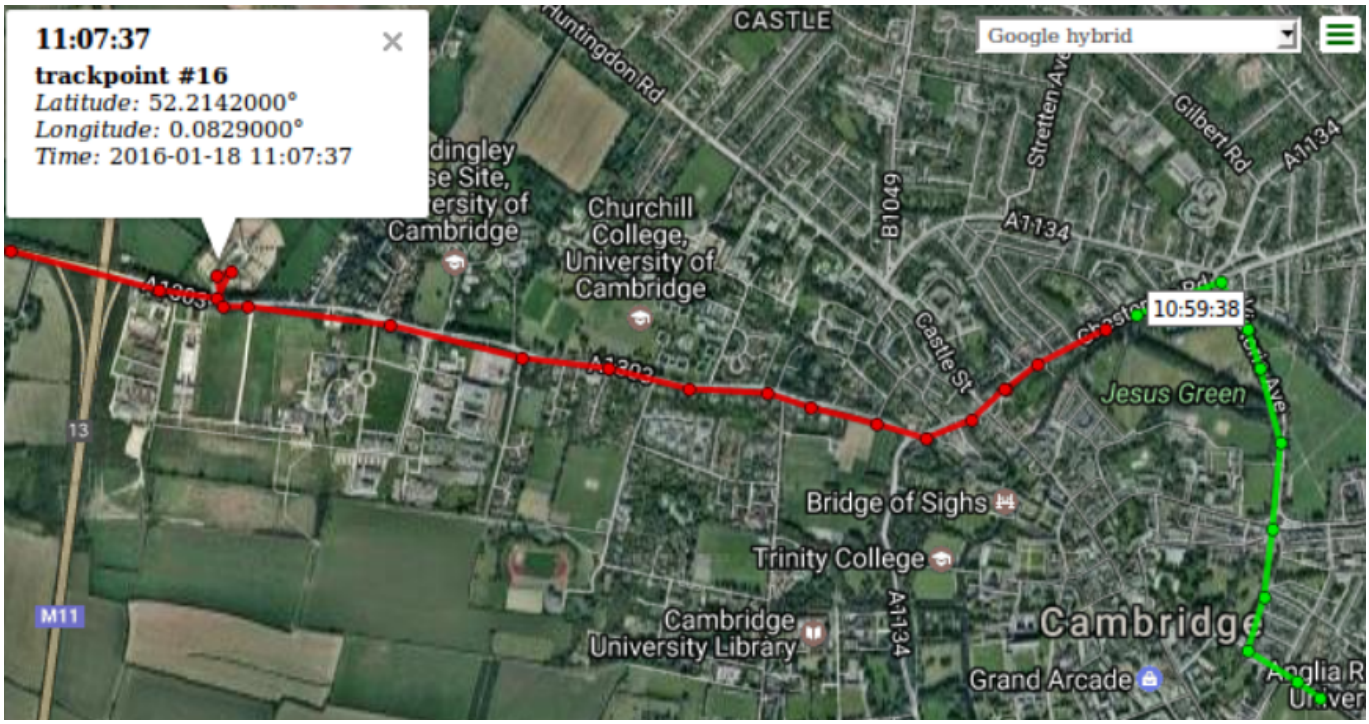
I make the following conclusions:

- 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.

4.3.3 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. Detailed evaluation results are given in the appendix.

I now apply the optimisation nr. 1 on the same test set. 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. 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. The second optimisation takes into account only equally congested historical trips. 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.

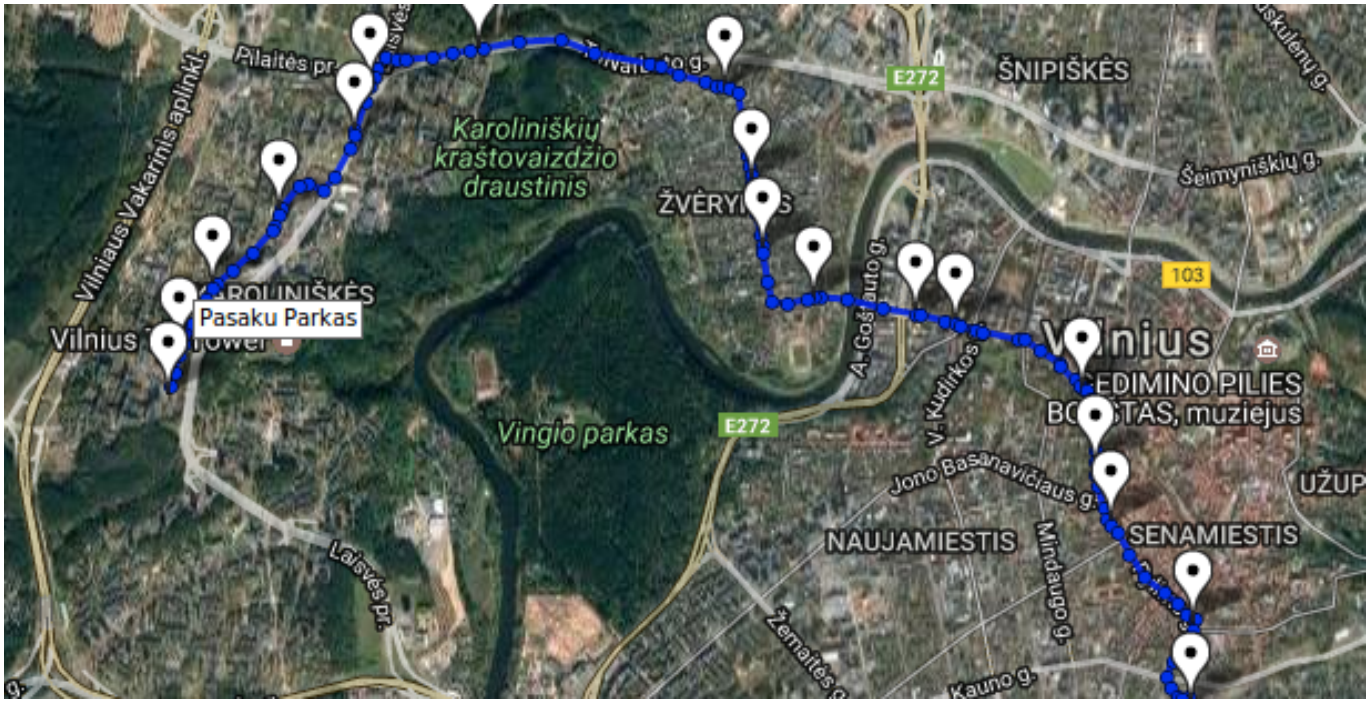
Comparison with the Timetable

The prediction system is useful only if for a particular route it is more accurate than the timetable. If I know when the bus is expected to arrive according to the timetable, I can compare whether my algorithm performs better than the timetable or not.

A company named Traffi targets the same arrival time prediction problem as me. The company agreed to share their traffic data. I was given the November 2016 GPS data. The Traffi data shows when the bus arrives at the stop **and also** when the bus is expected to arrive according to the timetable. This data easily allows checking how well my algorithm performs compared to the timetable:

Assume the arrival time prediction is what the timetable announces. One can view the timetable as just an another prediction algorithm and compute the MAE values for this algorithm as usual. I can 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.

I merge two tables of errors and present them together. 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. The bus is far away from the last stop at the beginning of the route. Also, 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, the prediction algorithm outperforms the timetable starting from the later stops.

		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

4.4 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 1051308-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	93	699	567	175
Bus Station	ATT	979	1649	1672	971
From stop nr. 2	MAE	117	153	392	109
St Pauls Square	ATT	846	898	1000	802
From stop nr. 3	MAE	130	125	319	78
Kingsway Link	ATT	696	703	772	665
From stop nr. 4	MAE	126	121	273	73
Borough Hall	ATT	634	640	716	592
From stop nr. 5	MAE	83	97	163	71
Bedford Hospital We	ATT	536	527	585	486
From stop nr. 6	MAE	68	71	151	34
Whitbread Avenue	ATT	401	390	427	366
From stop nr. 7	MAE	82	52	64	48
The Keep	ATT	331	324	326	316
From stop nr. 8	MAE	34	34	18	18
Spring Road	ATT	200	224	197	215
From stop nr. 9	MAE	25	29	12	23
Margetts Road	ATT	140	160	132	159
Last stop: Stop H - Church St		7 trips used	81 trips used	11 trips used	28 trips used

Chapter 5

Conclusion

Traffic analysis is a hot topic nowadays. In this project I had a chance to explore what it means to deal with a large amount of GPS data. 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.

5.1 Future Work

Below is the list of things that I believe can improve the project most:

- Get better GPS data. I worked with the data where the new point is received after 30s. A lot of things can change in 30s. For example, a bus can arrive at the bus stop, briefly stop at it and depart in 30s. In that case the GPS data will not indicate that the bus stopped at the bus stop. Thus, I can not be sure at what exact time the bus actually arrived at the stop. This complicates the arrival time prediction evaluation.

  There is another problem with the current GPS data. Sometimes the GPS device gets "stuck". For a few minutes it reports the same bus location. After that there is a big change between the last and new (latitude, longitude) pairs. Finding a way to fix this would help.

- Get better route data. The routes I have are outdated. Some buses do not follow any of the routes present in my set.

- Make the real-time system process the data faster. Since I did not spend much time for this optional part, there is plenty of room to improve it. Currently I do not predict the arrival time for **each** bus in 30s. One performance bottleneck is the $FollowsPath$ predicate implementation. I am using it often to make sure that the bus both follows the path and has not deviated from it. I think it is possible to reduce the time complexity from $O(nm)$ to $O(n+m)$, where n is the number of recent sub-trip points, m is the number of historical path points. This reduction would allow me to process significantly more buses in 30s.

  Another thing worth considering is to parallelise the system. My system is built in a way that for each bus I independently predict its arrival time to future stops. In the best case (if the lab has enough computation resources), one can consider dedicating a separate core per bus.

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 1002879-20150601-20150724		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	100	109	170	53
School for Boys	ATT	600	636	678	532
From stop nr. 2	MAE	95	93	158	59
Lutterworth Road	ATT	564	593	640	503
From stop nr. 3	MAE	82	84	154	50
St Andrews Hospital	ATT	495	539	578	453
From stop nr. 4	MAE	52	58	79	35
Palmerston Road	ATT	352	410	398	342
From stop nr. 5	MAE	53	50	78	29
Cliftonville	ATT	310	368	357	300
From stop nr. 6	MAE	45	51	69	25
Alexandra Road	ATT	248	300	284	229
Last stop: Northampton Draper		14 trips used	59 trips used	23 trips used	20 trips used

Route 1054060-20150531-22000909		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	150	518	1008	387
Central Railway Station	ATT	3668	4910	4644	4080
From stop nr. 2	MAE	150	339	835	284
The Point	ATT	3295	3642	4053	3600
From stop nr. 3	MAE	157	280	566	267
Lasborough Road	ATT	2626	2870	3216	2906
From stop nr. 4	MAE	179	175	240	135
Simpson Close	ATT	1231	1334	1369	1360
From stop nr. 5	MAE	174	125	170	189
Luton Stn Stand 3	ATT	709	648	697	729
Last stop: Airport Bus Station		10 trips used	63 trips used	28 trips used	22 trips used

Route 991943-20150412-20411231		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	221	118	114	172
Arundel Road	ATT	1176	1096	1031	901
From stop nr. 2	MAE	201	109	110	139
Kingsway	ATT	1114	1020	994	846
From stop nr. 3	MAE	191	97	118	141
Lincoln Road	ATT	951	201	898	747
From stop nr. 4	MAE	193	89	130	126
Birch Link	ATT	812	751	781	645
From stop nr. 5	MAE	202	86	130	125
Kenilworth Road	ATT	729	650	675	572
From stop nr. 6	MAE	132	72	113	117
Sainsburys	ATT	566	502	537	480
From stop nr. 7	MAE	130	71	106	117
Liverpool Road	ATT	435	394	408	413
From stop nr. 8	MAE	102	69	100	113
Dunstable Place	ATT	348	323	327	338
From stop nr. 9	MAE	102	63	99	134
Flowers Way	ATT	279	247	267	272
From stop nr. 10	MAE	102	60	99	115
Crown Court	ATT	210	163	189	218
Last stop: Church Street		16 trips used	80 trips used	9 trips used	13 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	135	121	141	136
Newland Centre	ATT	640	664	632	603
From stop nr. 2	MAE	45	39	43	38
Bonham Court	ATT	292	305	271	268
From stop nr. 3	MAE	36	34	26	29
Havelock Street	ATT	217	231	202	205
From stop nr. 4	MAE	32	26	18	37
Lobelia Road	ATT	166	186	164	158
From stop nr. 5	MAE	20	21	21	35
Walnut Crescent	ATT	101	115	103	115
Last stop: Aster Road		25 trips used	166 trips used	23 trips used	10 trips used

Route 1046521-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	178	116	111	156
Station Road	ATT	2500	2326	2258	2293
From stop nr. 2	MAE	163	125	107	145
St Ives P & R	ATT	2390	2218	2154	2193
From stop nr. 3	MAE	150	118	105	144
Fen Drayton Lakes	ATT	2117	1941	1960	1973
From stop nr. 4	MAE	155	111	102	138
Swavesey	ATT	1970	1803	1833	1832
From stop nr. 5	MAE	140	99	101	144
Longstanton P & R	ATT	1705	1554	1585	1568
From stop nr. 6	MAE	137	91	92	133
Oakington	ATT	1441	1321	1345	1305
From stop nr. 7	MAE	116	86	80	130
Histon & Impington	ATT	1178	1083	1106	1055
From stop nr. 8	MAE	94	76	77	101
Regional College	ATT	955	889	898	841
From stop nr. 9	MAE	80	71	68	91
Science Park	ATT	842	787	799	747
From stop nr. 10	MAE	34	30	32	34
Central Cambridge	ATT	229	244	231	207
		49 trips used	39 trips used	17 trips used	55 trips used

Route 711130-20140127-22000909		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	225	130	179	71
Harpur Street	ATT	593	695	705	1100
From stop nr. 2	MAE	47	37	29	25
Union Street	ATT	146	156	164	102
Last stop: Linden Road		18 trips used	73 trips used	13 trips used	5 trips used

Route 1047079-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	309	118	114	172
Trumpington P & R	ATT	826	1096	1031	901
From stop nr. 10	MAE	323	60	99	115
Foster Road	ATT	571	163	189	218
Last stop: Railway Stn Stop 9		126 trips used	126 trips used	104 trips used	159 trips used

Route 1050724-20150524-20150530		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	343	357	442	265
Milton Keynes Coach	ATT	1941	2108	2035	1762
From stop nr. 2	MAE	151	181	106	89
Chaucer Road	ATT	321	373	277	243
Last stop: Bus Station		20 trips used	123 trips used	42 trips used	40 trips used

Route 1050843-20150524-20150530		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	173	177	168	172
Milton Keynes Coach	ATT	1238	1208	1257	1265
From stop nr. 2	MAE	130	107	123	129
The Point	ATT	639	572	664	673
Last stop: Central Railway Station		126 trips used	501 trips used	148 trips used	241 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	190	218	178	228
Luton Stn Stand 10	ATT	1426	1473	1435	1354
From stop nr. 2	MAE	147	187	151	213
Clifton Rd West	ATT	1086	1192	1096	1067
From stop nr. 3	MAE	127	170	136	161
Stanton Rd West	ATT	794	900	814	747
From stop nr. 4	MAE	123	163	127	153
White Lion West	ATT	528	633	532	479
From stop nr. 5	MAE	119	163	138	153
CB College	ATT	480	583	475	426
From stop nr. 6	MAE	123	156	102	154
Stop P - Court Dr	ATT	383	486	357	344
From stop nr. 7	MAE	88	124	73	147
Stop M - High St	ATT	214	260	190	229
Last stop: Stop H - Church St		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	450	242	278	534
Luton Stn Stand 11	ATT	3841	3521	3466	3832
From stop nr. 2	MAE	418	232	247	548
Galaxy Centre	ATT	3661	3354	3333	3648
From stop nr. 3	MAE	330	194	132	439
Halfway Avenue	ATT	2931	2564	2577	2920
From stop nr. 4	MAE	201	140	109	225
Brinklow Roundabout	ATT	1199	1167	1114	1171
From stop nr. 5	MAE	106	114	105	108
The Point	ATT	564	590	596	551
Last stop: Central Railway Station		19 trips used	56 trips used	19 trips used	41 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	192	220	342	163
North Gate Bay 11	ATT	1216	1320	1481	1171
From stop nr. 2	MAE	115	113	151	102
Abington Square	ATT	924	990	1013	877
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	61	70	83
The Square	ATT	782	720	735	744
From stop nr. 2	MAE	85	46	37	49
Police Station	ATT	642	595	608	618
Last stop: The Square		7 trips used	62 trips used	31 trips used	48 trips used

Route 1026152-20150601-20150717		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	219	188	171	76
George Str Stop D	ATT	1109	994	985	850
Last stop: RS Components		69 trips used	118 trips used	26 trips used	15 trips used

Route 1030740-20150601-20151231		The morning rush hour	The working hourse evening rush ho		The night time
From stop nr. 1	MAE	97	100	114	124
Stop G - Church St	ATT	824	800	845	866
From stop nr. 2	MAE	50	71	73	69
Stanton Rd East	ATT	466	481	482	492
From stop nr. 3	MAE	32	57	56	55
Clifton Rd East	ATT	220	241	239	248
From stop nr. 4	MAE	22	44	42	41
Shuttle Link	ATT	92	110	97	111
Last stop: Luton Stn Stand 1		75 trips used	186 trips used	66 trips used	97 trips used

Route 1047094-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	151	132	113	142
Railway Stn Stop 1	ATT	1115	1047	987	1106
From stop nr. 2	MAE	150	123	82	130
Foster Road	ATT	198	202	184	230
Last stop: Trumpington P & R		50 trips used	309 trips used	116 trips used	68 trips used

Route 1050847-20150524-20150530		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	170	170	225	231
Central Railway Station	ATT	1146	1256	1173	1182
From stop nr. 2	MAE	152	149	184	188
The Point	ATT	801	867	794	843
Last stop: Milton Keynes Coach		53 trips used	267 trips used	120 trips used	102 trips used

Route 1051308-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	93	699	567	175
Bus Station	ATT	979	1649	1672	971
From stop nr. 2	MAE	117	153	392	109
St Pauls Square	ATT	846	898	1000	802
From stop nr. 3	MAE	130	125	319	78
Kingsway Link	ATT	696	703	772	665
From stop nr. 4	MAE	126	121	273	73
Borough Hall	ATT	634	640	716	592
From stop nr. 5	MAE	83	97	163	71
Bedford Hospital We	ATT	536	527	585	486
From stop nr. 6	MAE	68	71	151	34
Whitbread Avenue	ATT	401	390	427	366
From stop nr. 7	MAE	82	52	64	48
The Keep	ATT	331	324	326	316
From stop nr. 8	MAE	34	34	18	18
Spring Road	ATT	200	224	197	215
From stop nr. 9	MAE	25	29	12	23
Margetts Road	ATT	140	160	132	159
Last stop: Stop H - Church St		7 trips used	81 trips used	11 trips used	28 trips used

Route 992088-20150526-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	231	117	114	172
Arundel Road	ATT	1170	1094	1031	901
From stop nr. 2	MAE	208	108	110	139
Kingsway	ATT	1108	1018	994	846
From stop nr. 3	MAE	194	96	118	141
Lincoln Road	ATT	994	899	898	747
From stop nr. 4	MAE	193	88	130	126
Birch Link	ATT	804	749	781	645
From stop nr. 5	MAE	199	85	130	125
Kenilworth Road	ATT	719	648	675	572
From stop nr. 6	MAE	122	71	113	117
Sainsburys	ATT	551	501	537	480
From stop nr. 7	MAE	116	70	106	117
Liverpool Road	ATT	418	393	408	413
From stop nr. 8	MAE	88	68	100	119
Dunstable Place	ATT	332	321	327	338
From stop nr. 9	MAE	90	62	99	134
Flowers Way	ATT	264	245	267	272
From stop nr. 10	MAE	89	59	99	115
Crown Court	ATT	194	161	189	218
Last stop: Church Street		15 trips used	80 trips used	9 trips used	13 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	93	63	74	84
33 Park Drive	ATT	378	288	311	413
From stop nr. 2	MAE	94	63	72	91
South Oval Shops	ATT	375	282	295	402
From stop nr. 3	MAE	47	30	29	32
Witham Way	ATT	187	167	171	172
From stop nr. 4	MAE	28	26	28	19
Woodside Walk	ATT	126	117	123	125
Last stop: Witham Green		37 trips used	177 trips used	38 trips used	49 trips used

Route 1030920-20150601-20151231		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	298	314	360	537
Arundel Road	ATT	1206	1216	1203	1285
Last stop: Laxton Close		68 trips used	162 trips used	41 trips used	13 trips used

Route 1054599-20150531-22000909		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	586	496	311	1285
Bus Station	ATT	2372	2164	2048	3028
From stop nr. 2	MAE	438	371	256	483
Chaucer Road	ATT	1842	1758	1694	1999
Last stop: Milton Keynes Coach		44 trips used	177 trips used	46 trips used	94 trips used

Route 1050762-20150524-20150530		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	214	180	351	181
Milton Keynes Coach	ATT	2601	2696	2794	2675
From stop nr. 2	MAE	165	141	300	132
The Point	ATT	1950	2114	2206	2089
From stop nr. 3	MAE	141	144	229	122
Central Railway Station	ATT	1488	1563	1596	1600
Last stop: High Street		60 trips used	306 trips used	78 trips used	154 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	158	255	201	159
Trumpington P & R	ATT	822	867	747	860
From stop nr. 2	MAE	80	192	130	65
Porson Road	ATT	409	529	415	361
From stop nr. 3	MAE	63	128	105	38
Nuffield Hospital	ATT	303	384	324	271
From stop nr. 4	MAE	37	43	51	27
Leys School	ATT	185	194	184	167
Last stop: Pembroke Street		43 trips used	79 trips used	14 trips used	27 trips used

Route 1046531-20150531-20150830		The morning rush hour	The working hours	The evening rush hour	The night time
From stop nr. 1	MAE	228	160	181	223
Longstanton P & R	ATT	2143	1946	1993	1931
From stop nr. 2	MAE	221	154	194	214
Oakington	ATT	1920	1746	1792	1708
From stop nr. 3	MAE	213	150	185	209
Histon & Impington	ATT	1662	1510	1547	1452
From stop nr. 4	MAE	198	142	173	186
Regional College	ATT	1435	1318	1343	1249
From stop nr. 5	MAE	190	134	165	179
Science Park	ATT	1330	1214	1243	1152
From stop nr. 6	MAE	141	110	121	127
Central Cambridge	ATT	718	699	674	631
From stop nr. 7	MAE	129	96	111	116
Regent Street	ATT	524	485	495	462
Last stop: Railway Stn Stop 1		166 trips used	253 trips used	70 trips used	185 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)