Marius Latinis

Bus Arrival Time Prediction

Computer Science Tripos – Part II

Christ's College

March 20, 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 an algorithm which given the most recent GPS data predicts when the bus will arrive at the future stops. Evaluate the algorithm and show that on average it predicts with error less than 80s.

Work Completed

All that has been completed appears in this dissertation.

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 March 20, 2017

Contents

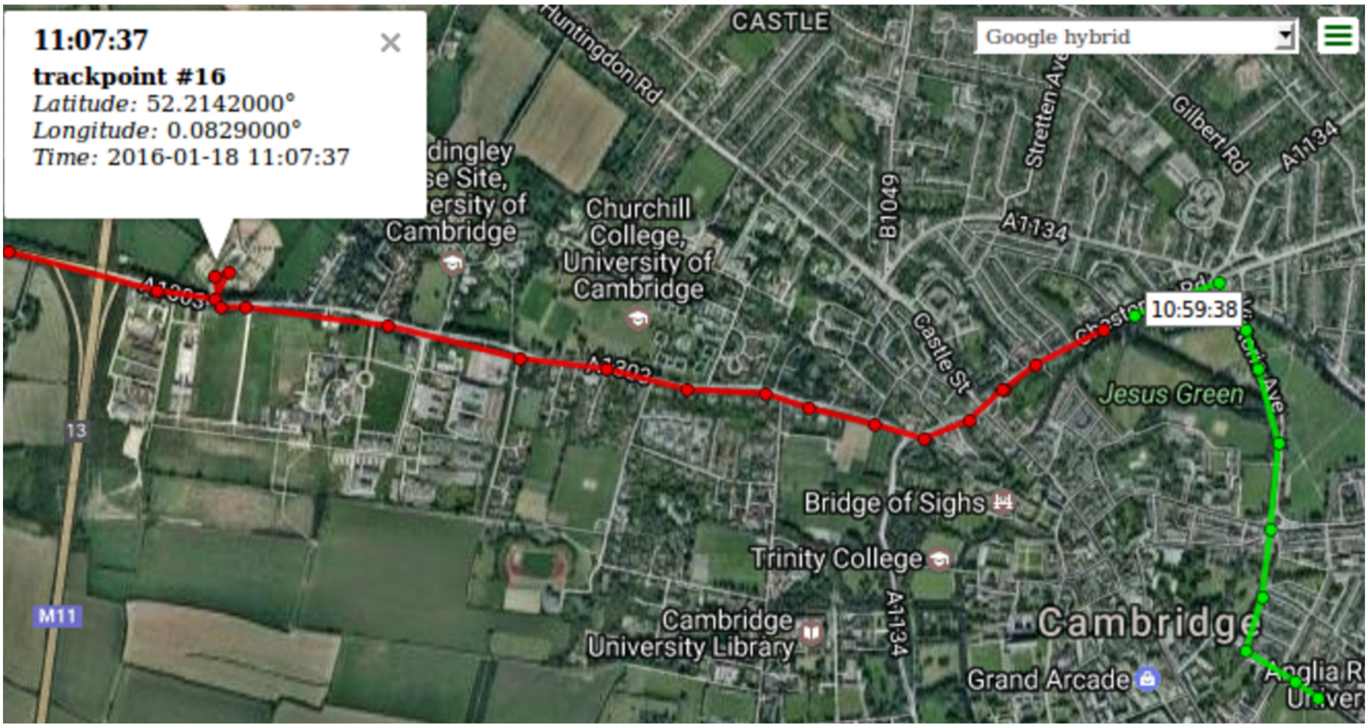
Chapter 1

Introduction

1.1 Motivation

In European countries public buses is a popular form of transportation. However, buses often arrive later than the timetable announces. Therefore, the project aims to build an algorithm which accuratelly predicts the bus arrival times based on the most recent GPS data.

1.2 Actual Problem Overview



11:07:37
trackpoint #16
Latitude: 52.2142000°
Longitude: 0.0829000°
Time: 2016-01-18 11:07:37

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

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

Chapter 2

Preparation

2.1 Starting Point

TODO(ml693): put something here

2.2 Preliminary Work

TODO(ml693): put something here

Chapter 3

Implementation

TODO(ml693): replace not my work by my work. The reason why I am not deleting implementation template yet is because it contains nice latex code.

3.1 Verbatim text

Verbatim text can be included using `\begin{verbatim}` and `\end{verbatim}`. I normally use a slightly smaller font and often squeeze the lines a little closer together, as in:

```
GET "libhdr"

GLOBAL { count:200; all  }

LET try(ld, row, rd) BE TEST row=all
                        THEN count := count + 1
                        ELSE { LET poss = all & ~(ld | row | rd)
                               UNTIL poss=0 DO
                               { LET p = poss & -poss
                                 poss := poss - p
                                 try(ld+p << 1, row+p, rd+p >> 1)
                               }
                             }
LET start() = VALOF
{ all := 1
  FOR i = 1 TO 12 DO
  { count := 0
    try(0, 0, 0)
    writef("Number of solutions to %i2-queens is %i5*n", i, count)
    all := 2*all + 1
  }
  RESULTIS 0
}
```

3.2 Tables

Here is a simple example[1] of a table.

Left Justified	Centred	Right Justified
First	A	XXX
Second	AA	XX
Last	AAA	X

There is another example table in the proforma.

[1] A footnote

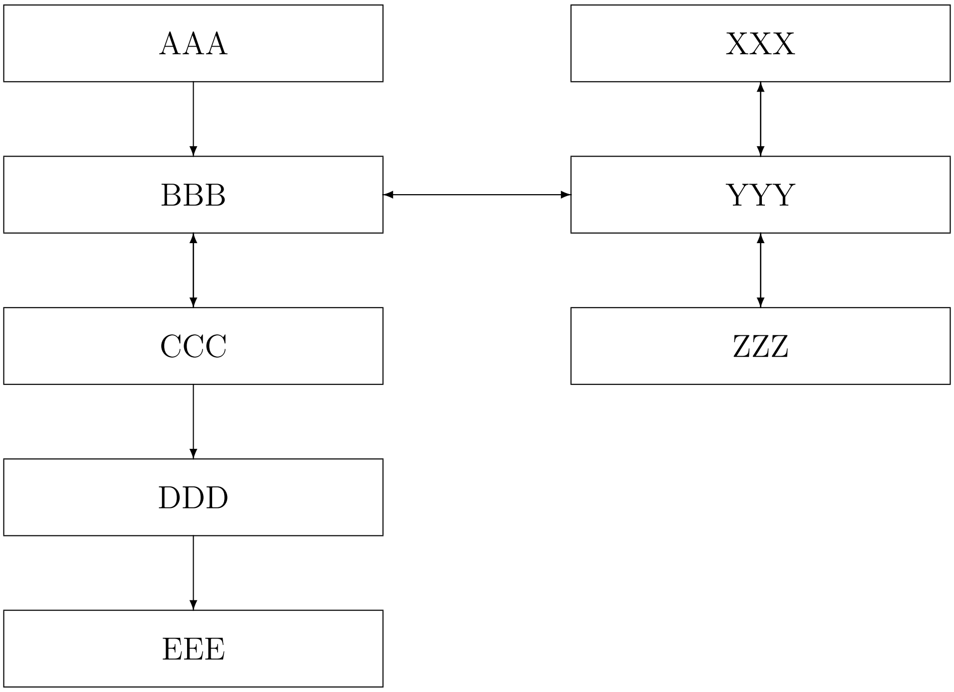


Figure 3.1: A picture composed of boxes and vectors.

3.3 Simple diagrams

Simple diagrams can be written directly in LaTeX. For example, see figure 3.1 on page 7 and see figure 3.2 on page 8.

3.4 Adding more complicated graphics

The use of LaTeX format can be tedious and it is often better to use encapsulated postscript (EPS) or PDF to represent complicated graphics. Figure 3.3 and 3.5 on page 9 are examples. The second figure was drawn using `xfig` and exported in `.eps` format. This is my recommended way of drawing all diagrams.

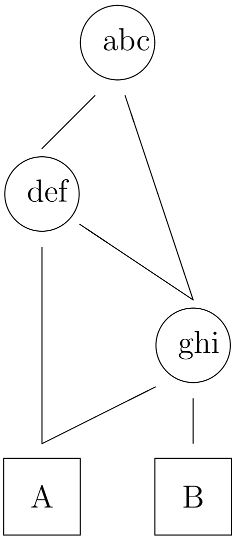


Figure 3.2: A diagram composed of circles, lines and boxes.

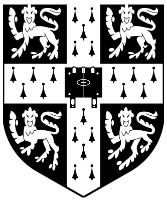


Figure 3.3: Example figure using encapsulated postscript

Figure 3.4: Example figure where a picture can be pasted in

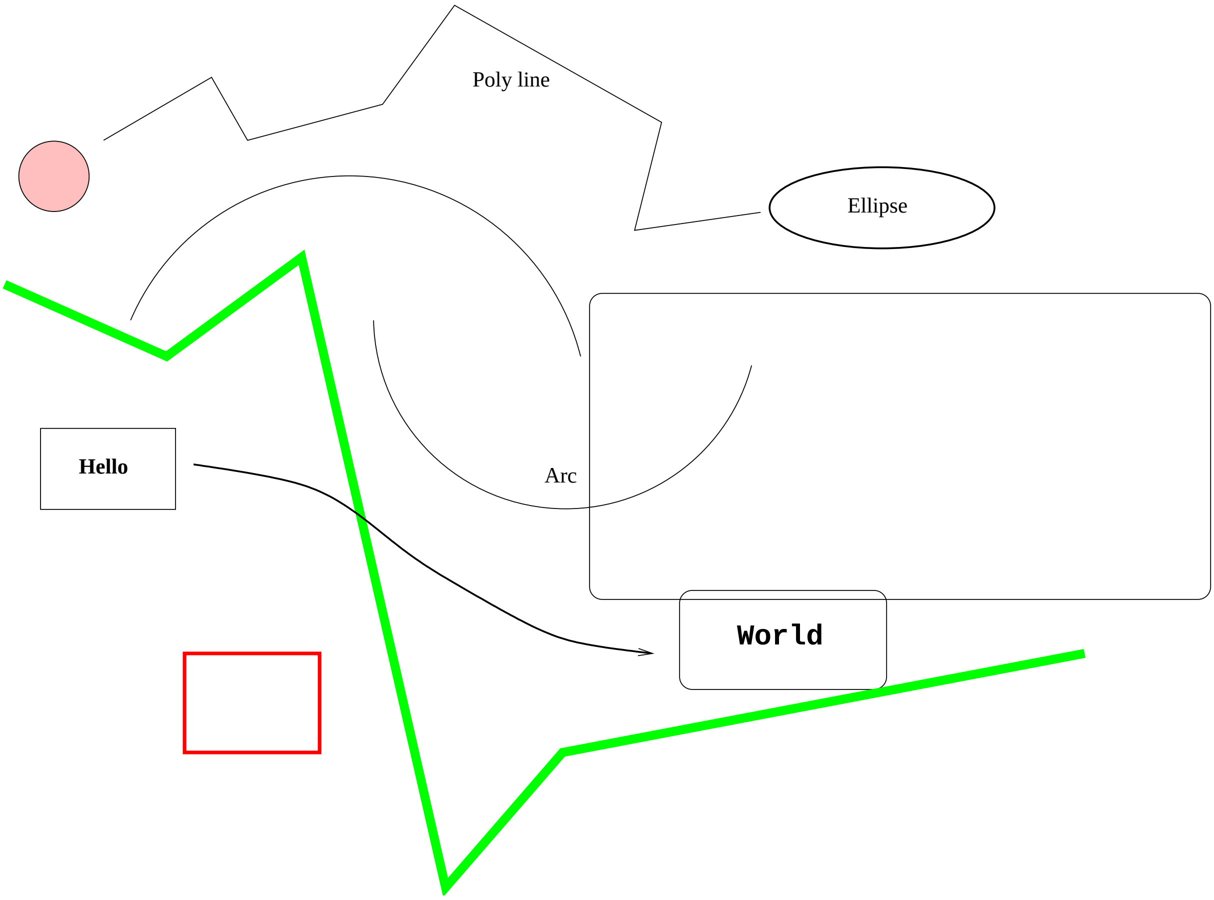


Figure 3.5: Example diagram drawn using `xfig`

Chapter 4

Evaluation

4.1 Printing and binding

Use a “duplex” laser printer that can print on both sides to print two copies of your dissertation. Then bind them, for example using the comb binder in the Computer Laboratory Library.

4.2 Further information

See the Unix Tools notes at
`http://www.cl.cam.ac.uk/teaching/current-1/UnixTools/materials.html`

Chapter 5

Conclusion

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

If started again, I would have worked in exactly the same way.

Appendix A

Latex source

A.1 diss.tex

A.2 proposal.tex

```
% Note: this file can be compiled on its own, but is also included by
% diss.tex (using the docmute.sty package to ignore the preamble)
\documentclass[12pt,a4paper,twoside]{article}
\usepackage[pdfborder={0 0 0}]{hyperref}
\usepackage[margin=25mm]{geometry}
\usepackage{graphicx}
\usepackage{parskip}
\begin{document}

\begin{center}
\Large
Computer Science Tripos -- Part II -- Project Proposal\\[4mm]
\LARGE
How to write a dissertation in \LaTeX\\[4mm]

\large
M.~Richards, St John's College

Originator: Dr M.~Richards

14 October 2011
\end{center}

\vspace{5mm}

\textbf{Project Supervisor:} Dr M.~Richards

\textbf{Director of Studies:} Dr M.~Richards

\textbf{Project Overseers:} Dr F.~H.~King  \& Dr A.~W.~Moore

% Main document

\section*{Introduction}

\emph{The problem to be addressed.}

Many students write their CST dissertations in \LaTeX\ -- and spend a
fair amount of time learning just how to do that. The purpose of this
project is to write a demonstration dissertation that provides
a starting point to show how it is done.

This core proposal document will be augmented by a separately-printed
cover sheet at the front and a resource form at the end. Additional
sheets for risk assessment and human resources may also need to be
included.
```

```
This document will elaborate much of the material that is summarised on
the additional sheets.

\section*{Starting point}

\emph{Describe existing state of the art, previous work in this area,
  libraries and databases to be used. Describe the state of any
  existing codebase that is to be built on.}

I am already able to write prose using the English language. I have an
online dictionary, etc.

\section*{Resources required}

\emph{A note of the resources required and confirmation of access.}

For this project I shall mainly use my own quad-core computer that
runs Fedora Linux. Backup will be to github and/or to an SVN
repository on an external hard disk that is dumped to writable CD/DVD
media. I have another similar computer to hand should my main machine
suddenly fail. I require no other special resources.

\section*{Work to be done}

\emph{Describe the technical work.}

The project breaks down into the following sub-projects:

\begin{enumerate}

\item The construction of a skeleton dissertation with the required
  structure. This involves writing the Makefile and making dummy
  files for the title page, the proforma, chapters 1 to 5, the
  appendices and the proposal.

\item Filling in the details required in the cover page and proforma.

\item Writing the contents of chapters 1 to 5, including examples of
  common \LaTeX\ constructs.

\item Adding a example of how to use floating figures and ``encapsulated
  PostScript'' or PDF diagrams.

\end{enumerate}

\section*{Success citeria}

\emph{Describe what you expect to be able to demonstrate at the
end of the project and how you are going to evaluate your achievement.}

The project will be a success if I have a completed dissertation with
the correct chapter titles and I have achieved my other success
criteria, which are to blah \ldots


\section*{Possible extensions}

{\em Potential further envisaged evaluation metrics or extensions.}

If I achieve my main result early I shall try the following
alternative experiment or method of evaluation \ldots


\section*{Timetable}

\emph{A workplan of perhaps ten or so two-week work-packages,
as well as milestones to be achieved along the way. Provide a
target date for each milestone.}

Planned starting date is 16/10/2011.

\begin{enumerate}
```

```
\item \textbf{Michaelmas weeks 2--4} Learn to use X. Read book Y. Read papers Z.

\item \textbf{Michaelmas weeks 5--6} Do preliminary test of Q.

\item \textbf{Michaelmas weeks 7--8} Start implementation of main task A.

\item \textbf{Michaelmas vacation} Finish A and start main task B.

\item \textbf{Lent weeks 0--2} Write progress report. Generate corpus of
  test examples. Finish task B.

\item \textbf{Lent weeks 3--5} Run main experiments and achieve working project.

\item \textbf{Lent weeks 6--8} Second main deliverable here.

\item \textbf{Easter vacation:} Extensions and writing dissertation main
  chapters.

\item \textbf{Easter term 0--2:}  Further evaluation and complete dissertation.

\item \textbf{Easter term 3:} Proof reading and then an early submission
  so as to concentrate on examination revision.

\end{enumerate}

\end{document}
```

Appendix B

Makefile

B.1 makefile

B.2 refs.bib

```
@BOOK{Lamport86,
TITLE = "{LaTeX} --- a document preparation system --- user’s guide
and reference manual",
AUTHOR = "Lamport, L.",
PUBLISHER = "Addison-Wesley",
YEAR = "1986"}

@REPORT{Moore95,
TITLE = "How to prepare a dissertation in LaTeX",
AUTHOR = "Moore, S.W.",
YEAR = "1995"}
```

Appendix C

Project Proposal

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 $\beta$ streaming GPS data:
    a) Extract $l_1$ - the current location of $\beta$
    b) Look at $\beta$ route to see the next locations $l_2$, …, $l_n$
    // The (c) part will be implemented following the summation idea above
    c) Predict $\beta$ arrival times $t_2$, …, $t_n$ for each $l_2$, ..., $l_n$ and store these predictions
            in a multimap data structure Bus[$\beta$] Arrives{($l_2$, $t_2$), ... ($l_n$, $t_n$) }
```

4. <u>Prediction algorithm evaluation.</u> 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=Ml2XsklclPBf9ChuR7ZdJLX2Qmc&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)