Integrating Traffic Data Stream with Public Route Planning Algorithm and Personalizing it for the End Users

Soham Sinha
University of Alberta
Edmonton, AB, Canada
soham@ualberta.ca

Sankalp Prabhakar
University of Alberta
Edmonton, AB, Canada
sankalp@ualberta.ca

ABSTRACT

Route suggestions for public transportation has been integrated to different mapping services such as Google Maps or Bing Maps. However, those suggestions are often found to be unreliable for the end-users, mainly because of two reasons. Firstly, these route-finding techniques do not consider traffic conditions on the roads. Secondly, routes are generalized for every user, completely ignoring their commuting patterns.

We propose an architecture with a modified routing algorithm that aims to solve both the problems. Our system incorporates both real-time and historical traffic-data into public-transit routing framework. We also personalize transit routes for users by analyzing their commuting behavior. By experiments, we show that our system significantly improves the quality of route-suggestions compared to state-of-the-art techniques with minimal overhead (as low as 20 ms for a 3 km long route).

1. INTRODUCTION

Public transportation is one of the important means of communications around the globe. Wider adoption of public buses, trains etc. is encouraged by the governments for a number of reasons. Increasing use of public transport can help to alleviate environmental problems, for example by efficient usage of fuels. There are efforts in the research community as well to promote public transit. As part of that ongoing effort, a standard format is designed to represent the public transportation, its schedule and related information. This standard is named General Transit Feed Specification (GTFS) [2]. GTFS-formatted data has enabled software systems to manipulate public transit data efficiently. The standardization of public transportation data in the form of GTFS feed has opened up numerous possibilities of visualizations, optimizations and improvements related to public transit [8]. Many companies which provide map-related services, have adopted GTFS feed into their system [18, 23, 3, 5]. Most of these companies give public routing suggestions through various interfaces (online, mobile application etc.), mostly following Dijkstra's Algorithm for shortest path [11]. Our work revolves around public routing suggestions and its two main limitations. Firstly, the current implementations of public routing suggestions totally ignore the traffic conditions of roads. Secondly, the suggestions are generalized and same for every users. They don't account for users' commuting behavior in terms of their walking or cycling speeds. Because of these two drawbacks, public transit routing suggestions are often not accurate. We want to mitigate these two limitations to improve the routing suggestions.

Real-time and historical traffic conditions of roads have already been well studied in the literature [14, 22, 6, 20, 31, 21, 19, 24]. Some industry-standard applications [18, 3] show real-time and historical traffic conditions in their software systems as well. However, the traffic information both real-time and predicted, has only been used for private car routing suggestions. This has been largely ignored in public routing in all the applications that we know of. This makes public routing suggestions unreliable.

The improvement of route suggestions in public transit is important because it can increase the reliability in public transport among people. Reliance on public transportation translates into many benefits as described earlier. As the current routing suggestions for public transits do not consider delays due to traffic, one may easily get delayed reaching his/her destination taking the public buses. This is really inconvenient for the end-users. Another common issue is that the state-of-the-art applications do not take into account the different walking-speeds for various users; rather, they just provide a generalized result for every user. To the best of our knowledge, we are the first to propose a system that can integrate both real-time and historical traffic feed along with the end users' commuting pattern for routing suggestions in a public transportation system.

To briefly put, our contributions are:

1. Analyzing real-time and historical traffic data and integrating traffic-speed into the public routing algorithm.
2. Analyzing the end users' commuting pattern and deriving users' average walking-speed to provide personalized suggestions of public-transit routes.

The paper is organized as follows. In the next section, we describe the relevant works and previous research done related to this problem. In Section 3, we formally describe our problem. We demonstrate the architectural overview of our solution in Section 4. Some of the implementation details are mentioned in Section 5. Section 6 demonstrates our implementation for future traffic prediction and how it integrates with our system. In Section 7, we present the conducted experiments and compare with the existing techniques. Finally, we conclude and discuss future works in the last section.

2. RELATED WORK

GTFS [2] was standardized around 2009 by the community of public transit agencies, developers with primary help from Google. Since then, many applications [18, 23, 3, 5] have integrated it to their systems to suggest public routes. Google Maps [18], being the frontier of all these applications, suggests public transportation routes to its web-based and mobile-based users. Google Maps and other applications collect data in the form of GTFS from various agencies around the globe and integrates it to their mapping system. However, they just work on the agency-provided data to give routing suggestions from a source location to a destination. They do not consider any probable delay due to traffic.

On the other hand, the recent development of mobile location tracking systems have helped various software systems to keep track of real-time traffic information on their servers. Google keeps track of the real-time traffic levels [15] by monitoring the users' mobile devices [16] and also using other services such as Waze [29]. Although this traffic information is utilized to suggest optimized routing for private cars, it has not been incorporated in the domain of public transportation. Part of our work aims to mitigate this challenge. Apart from real-time traffic information, there are previous research works [24, 31, 26, 10] which try to exploit the historical traffic data to predict future traffic-conditions. These are complementary to our work because a higher efficiency in traffic prediction will only improve our routing suggestions.

There has been research on real-time tracking of public transportation. Thiagarajan et. al. has shown how crowd can be utilized to track public buses, underground metro rails etc [27]. Bast et. al has recently shown how the real-time movement of public transportation system can be integrated and visualized in maps [7]. TransitApp [5] also shows real-time bus status in their mobile application. However, none of these approaches try to suggest routes considering the delay caused by traffic. In our opinion, the reason for not addressing this problem is the high time-complexity to process the large amount of real-time public transit data. It is also cost-expensive. For example, Edmonton Transit System (a public bus agency for the city of Edmonton) [13] recently planned to deploy GPS devices and other advance features on their buses by expending a hefty 13.9 million Canadian Dollars. Compared to these expensive solutions, our proposed system has low overhead in terms of time-complexity and is cost-effective as well.

The work which is most closely related to our paper, is done by Cuong [12]. He developed an algorithm, CrowdRoute which utilizes the real-time information on trip delays provided by various users. Our algorithm does not depend on any trip related information provided by crowd for a particular route. In contrast, our system utilizes the traffic information and personalized data (walking and cycling speed) to recalculate the estimated time for a given route. In this way, our system is more generic than CrowdRoute because of the latter's dependence on specific trip-information. Additionally, an efficient implementation of CrowdRoute needs extensive infrastructural set-up to receive updates from mobile users. On the contrary, our system uses the already available information and is more cost-effective.

3. PROBLEM DEFINITION

In this section we formally describe the problems in public route suggestions. The problem can be divided into two distinct parts: 1) Incorporating traffic feed in public route suggestions, 2) Incorporating end user's commuting behavior.

3.1 Incorporating Traffic Feed

Public transit routing suggestions consist of several trips in buses, trains or metro railways. The routing suggestions are based on the schedules provided by the public transportation agencies in form of GTFS. These agencies make a generalized schedule, ignoring the probable traffic on the roads. Even if they care for the probable road-traffic, they may not be accurate. As the traffic-information on the roads is mainly ignored, the delay due traffic is also not accounted for in the routing suggestions. This makes the suggestions imprecise. Our challenge is to incorporate the available traffic information of roads into the routing suggestions, so that we can make them more realistic.

3.2 Incorporating End Users' Commuting Behavior

One important aspect of public route suggestions is that they consist of steps which involve walking. For instance, a user must walk to a nearby bus-stop to begin his/her journey. In all the current state-of-the-art applications, the walking-speed is assumed to be same for every user. In real-life, this is certainly not the case. For example, a fairly old lady is most likely to have a different average walking speed than a young gentleman. Existing applications ignore this fact, and it leads to poor suggestions. Our challenge is to understand the commuting patterns of end users and utilize it in route-suggestion. Now-a-days, different kinds of sensors are pretty common in mobile phones. We can use the data related to user-movement recorded by the sensors and integrate it into the public routing algorithm.

The above problems in public transport are the motivations for our work. We try to resolve both the challenges and present a unified solution for public route suggestions.

4. ARCHITECTURE

We provide an overview of our architecture in this section. We have divided our system into certain interrelated generic components. The architecture is outlined in Figure 1.

Our system has an interface which takes the source, destination and departure time from the user. This has been shown in Figure 1 by the top most rectangular box, named "Application Interface" (A. Int). This component sends the above mentioned information to the core of our system, "Main-Frame" (MF) (middle box in Figure 1). MF then asks information to four different components in the back-end, the four boxes in the last row in the same figure. We will now describe each of these four components. We include only an overview of those components in this section. More details regarding the implementation of those components have been demonstrated in the next section.

4.1 Back-end Components

4.1.1 Accelerometric Sensor Data

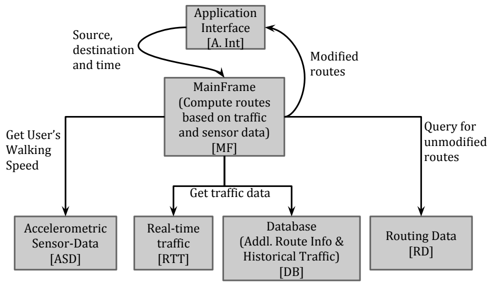

Application Interface [A. Int]
Source, destination and time
Modified routes
MainFrame (Compute routes based on traffic and sensor data) [MF]
Get User's Walking Speed
Query for unmodified routes
Get traffic data
Accelerometric Sensor-Data [ASD]
Real-time traffic [RTT]
Database (Addl. Route Info & Historical Traffic) [DB]
Routing Data [RD]

Figure 1: Architecture of our system

Accelerometric Sensor Data (ASD) has the analyzed data of user's commuting behavior. For our implementation, we have used Google Fit [17] service to get the average walking-speed of a user. We keep track of a user's walking speed for last one week. ASD provides the speed to the MF.

4.1.2 Real-time Traffic

Real-time traffic (RTT) component retrieves the real-time traffic data, in our case from a web-service. Real-time traffic is generated in custom format. We analyze that format and save the data in a structured way. We have used HERE Maps [3] Traffic Application Programming Interface (API) to get real-time traffic data. HERE Maps provides a JSON (or XML)-formatted data which we manually parse and store into (Java) classes. MF queries for the traffic-speed on certain road-segments to RTT. The RTT component filters relevant data and returns the speed back to MF.

4.1.3 Database

Database (DB) component contains the historically stored traffic data and any additional details of various routes in a given city. This component can be designed locally or can be implemented as a service like HERE Maps Traffic API. Additional routing details are also stored in database because of probable limitations of the next component, Routing Data (RD).

4.1.4 Routing Data

Routing Data (RD) component takes the source, destination location and departure time. It supplies a generalized, unmodified routing suggestions to MF. The suggestions provided by RD are essentially the same as given by existing applications, such as Google Maps, Bing Maps or HERE Maps. MF takes the unmodified routing suggestion and processes it with results from other components to suggest more accurate routes.

4.2 Modified Route Calculation Algorithm

Here we describe our algorithm to compute the modified routes. This algorithm is a part of MainFrame or MF in our architecture. But first, let us look at a sample unmodified route suggestion. An example can help to understand our algorithm in a better way.

A sample route consists of several steps with time-stamp of the step, length of the step, time (in seconds) to complete the step, details of the step and medium of the step (by bus or by walking). Optionally, in case of a bus-rides, the step can also include the source bus-stop, the destination bus-stop or the intermediate bus-stops and lengths between them. An example is given in Figure 2. The example presents a sample routing result from 7708, 109 Street, Edmonton, AB, Canada to University of Alberta, Canada.

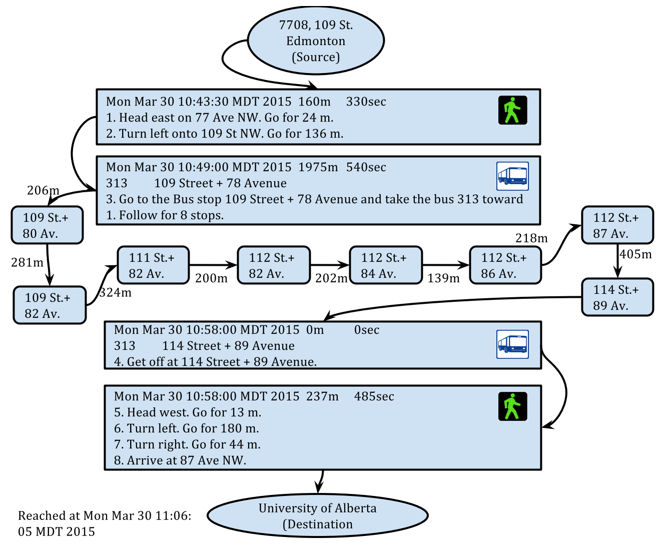

7708, 109 St. Edmonton (Source)
Mon Mar 30 10:43:30 MDT 2015 160m 330sec
1. Head east on 77 Ave NW. Go for 24 m.
2. Turn left onto 109 St NW. Go for 136 m.
Mon Mar 30 10:49:00 MDT 2015 1975m 540sec
313 109 Street + 78 Avenue
3. Go to the Bus stop 109 Street + 78 Avenue and take the bus 313 toward
1. Follow for 8 stops.
206m
109 St.+ 80 Av.
281m
109 St.+ 82 Av.
324m
111 St.+ 82 Av.
200m
112 St.+ 82 Av.
202m
112 St.+ 84 Av.
139m
112 St.+ 86 Av.
218m
112 St.+ 87 Av.
405m
114 St.+ 89 Av.
Mon Mar 30 10:58:00 MDT 2015 0m 0sec
313 114 Street + 89 Avenue
4. Get off at 114 Street + 89 Avenue.
Mon Mar 30 10:58:00 MDT 2015 237m 485sec
5. Head west. Go for 13 m.
6. Turn left. Go for 180 m.
7. Turn right. Go for 44 m.
8. Arrive at 87 Ave NW.
Reached at Mon Mar 30 11:06:05 MDT 2015
University of Alberta (Destination

Figure 2: Example of a public routing suggestion

There are total 8 steps (numbered) in the routing suggestion in Figure 2. For better representation and space-constraint, we have showed the routing result in a graphical form as grouped events. In our actual implementation, it is still in textual form. In Figure 2, each rectangular box represents a medium of transportation event. For example, the topmost rectangular box connected to the source is for a walking event. The first line in the rectangular box tells the time-stamp (24-hour format) at which the event(s) start, the length of the event(s) and the duration of it. The next lines are detailed information of the event. The steps in an event is numbered and inscribed inside the rectangular boxes. The topmost rectangular box has two steps in it. Starting time for the event is on Monday 30th March'15 at 10:43:30 in Mountain Daylight Time. Total length for those two steps is 160 meter, and duration is 330 seconds. The signs at the right-hand side of the rectangular boxes tell the mediums of the event. In case of the topmost box, it is walking. Next event which is connected to the topmost box, is a public bus-riding which starts at 10:49:00. The second line in case of public bus-riding event represents bus-number and bus-stop, separated by a white-space. In our example, the bus is Route 313, and user has to take the bus from "109 Street + 78 Avenue" bus-stop. The public bus-riding event is then connected with a series of rounded-rectangular boxes which represent the intermediate bus-stops. The edges between those boxes are labeled with the lengths between those bus-stops. For example, the length between the bus-stop "109 St. + 78 Av." and "109 St. + 80 Av." is 206 meters. The last bus-stop is connected to a public bus-riding event where the user needs to get off from the bus. This is how a route is interpreted. Now we will describe the algorithm through which we improve the unmodified routing result with the help of the data from other components of our architecture, given earlier in Figure 1.

Algorithm 1 presents the pseudo-code for our algorithm. The algorithm is extremely simple and has low overhead.

Algorithm 1 : Calculating the modified route considering traffic-feed

```
1  function FindRoute ( $Src$, $Dest$, $CurTime$ )
2      $WalkSpeed$ = getFromASD ()
3      $unmodifiedRoute$ = getFromRD ($Src$, $Dest$, $CurTime$, $WalkSpeed$)
4      $modifiedRoute$ = copy ($unmodifiedRoute$)  ▷ kept a copy of $modifiedRoute$ for comparison
5      for each $step$ in $modifiedRoute$ do
6          if $step.type$ = “by bus” then
7              $modifiedDuration = 0$
8              for each $stop$ in $steps.intermediateStops$ do
9                  $realSpeed$ = getFromRTT($stopNames$)
10                 $modifiedDuration$ += $stop.length$ / $realSpeed$
11             end for
12             $delay = modifiedDuration - step.duration$
13             modifyStep(step, delay)  ▷ this function modifies this $step$ and subsequent steps with the $delay$
14         end if
15     end for
16     return $modifiedRoute$
17 end function
```

We first get the walking speed from the user from the ASD module. Then, we get the unmodified routing result from the RD module. Important point to observe here is that we pass the walking speed of a user in this module. This is done because the unmodified result should consider the walking speed of the user and suggest route accordingly; then we can modify the route based on real-time traffic data. We check each steps in the routing result whether the step's medium is “by bus”. For that kind of a step, we iterate a loop over all stops within that bus journey. Inside the loop, we get the real-time speed of traffic between the bus-stops. Once we've the speed information, we calculate the modified duration accounting for the traffic. After measuring the delay, it is added to the concerned step and in all subsequent steps.

Because of the simplicity of our algorithm, it imposes no real overhead on the existing route calculation methodology. This is verified by experimentation, presented in Section 7. Although we have only shown real-time traffic feed in Algorithm 1, it can still be tweaked to account for the historical traffic patterns which could be useful in predicting future traffic. We have provided an explanation in Section 6 on how traffic profiling can be used to predict future traffic-level. There are previous research work which also talks about traffic prediction [24, 31, 26, 10]. These works can be hooked into our architecture (specifically, in the DB module).

5. IMPLEMENTATION DETAILS

In this section, we present some of the important implementation for various components mentioned in the architecture diagram in Figure 1.

5.1 Getting Users' Walking Speed

We have used Google Fit [17] to get the end users' walking speed. Users can install the Google Fit app on their android-based mobile devices to record their commuting patterns like number of steps, distance traveled or walking speed and store it in a central repository, called the Google Fitness Store (see Figure 3). We query this information once the users' have been authorized. We use oauth2, an open standard for authorization of the end users. An HTTP GET request is constructed using Google Fit APIs to query against the repository, which returns the users' walking speed, both instantaneous or average speed over a time interval.

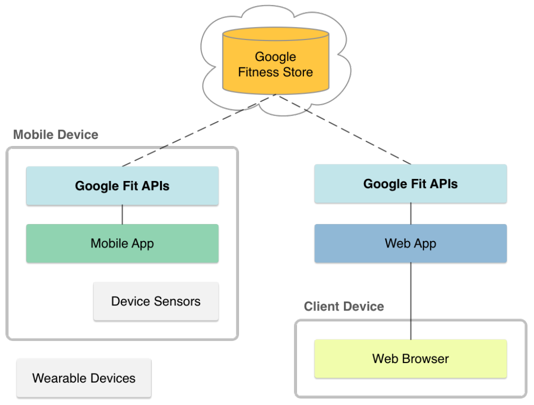


Figure 3: Google Fit Platform[1]

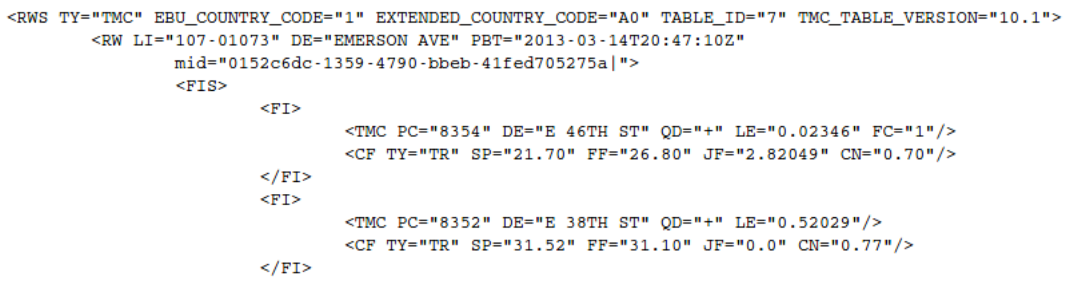
5.2 Fetching Unmodified Route

We have used HERE Maps Routing API [3] to get the routing suggestions. The HERE Maps Routing API calculates routes between two or more locations and provides additional route-related information. We construct an HTTP GET request using user's source, destination location, departure time-stamp and walking-speed as query parameters. Also, each request must include the authentication parameters (app_id and app_code) to access the resources of the API. Once the query is processed, we get a list of all possible routing suggestions between the source and destination location at a given time. This is similar to how other routing applications like Google or Bing Maps functions. We chose HERE Maps for our implementation to maintain consistency across route planning as we make use of HERE Traffic API to get the real-time traffic updates. The result of HERE Maps Routing API and Google Maps API for routing are same because they all use GTFS data provided by transit agencies.

5.3 Fetching Real-time Traffic-feed

We make use of HERE Maps Traffic API to get information about the traffic levels in a particular route in real-time. To obtain traffic data via the HERE Traffic API, it is necessary to formulate a request that combines the URL and a set of parameters to specify the required response (see URL below). One of the parameters used is the bounding box (bbox), which is created using the latitude, longitude coordinates for source and destination locations of a given route. Once the query is processed, the web service allows access to real-time traffic flow data in XML or JSON, including information on speed and congestion for the region(s) defined in each request. The service can deliver additional data such

[1]https://developers.google.com/fit/overview



```
<RWS TY="TMC" EBU_COUNTRY_CODE="1" EXTENDED_COUNTRY_CODE="A0" TABLE_ID="7" TMC_TABLE_VERSION="10.1">
        <RW LI="107-01073" DE="EMERSON AVE" PBT="2013-03-14T20:47:10Z"
                mid="0152c6dc-1359-4790-bbeb-41fed705275a|">
                <FIS>
                        <FI>
                                <TMC PC="8354" DE="E 46TH ST" QD="+" LE="0.02346" FC="1"/>
                                <CF TY="TR" SP="21.70" FF="26.80" JF="2.82049" CN="0.70"/>
                        </FI>
                        <FI>
                                <TMC PC="8352" DE="E 38TH ST" QD="+" LE="0.52029"/>
                                <CF TY="TR" SP="31.52" FF="31.10" JF="0.0" CN="0.77"/>
                        </FI>
```

Figure 4: Sample XML Response for Real-time Traffic

as the geometry of the road segments to which the flow data relates to (see Figure 4 for details).

A request matching the latitude and longitude of source and destination is formulated as follows:

```
http://traffic.cit.api.here.com/traffic/6.1/
flow.xml?app_id=DemoAppId01082013GAL
&app_code=AJKnXv84fjrb0KIHawS0Tg
&bbox=39.8485715,-86.096986;39.835893,-86.0757964
```

The XML response in Figure 4 to a request like the above one, gives the real-time traffic speed (SP) for different road segments in a particular route within the bounding box specified in the user query. It also contains some other valuable traffic parameters like the Free Flow (FF) speed, which is the average speed that a motorist would travel if there were no congestions or other adverse conditions like road accidents or climatic disruptions. Jam Factor (JF) provides insight about the congestion level on road segments, a higher value of JF is an indication of higher levels of traffic. We have used these parameters in our predictive model to forecast traffic conditions as well.

We have analyzed the XML response in Figure 4. The node named ”RW” represents a single roadway inside the given bounding-box. In our example, ”Emerson Ave” is the roadway for which the real-time speed is given. The child-node ”FI” inside parent ”RW” represents the crossing roadways. Speed and other information between the crossing roadways are given as attributes and child-nodes of ”FI”. For example, a crossing roadway on ”Emerson Ave” is ”E 38th St”. The free-flow speed and real-time speed between the previous crossing roadway and ”E 38th St” are 31.10 km/h and 31.52 km/h respectively.

The traffic speed (SP) mentioned above is recorded for private vehicles. We use a formula (see equation 1) to derive the real-time speed of public buses (RB) by making use of the scheduled bus-speed (SB) which is a fixed speed based on their defined schedules.

$$RB = (SP/FF) * SB \tag{1}$$

RB is the aligned speed in real-time traffic for any public transport. Once, we have the RB speed component, we can use it to find the actual delay for the all road segments for which the real-time traffic feed is available.

5.4 Historical Traffic-feed

We maintain a local database containing historical feed of traffic as well as the routing information about different routes, bus stops and schedules in a given city. Part of the historical data is synthetically generated to analyze and predict future traffic speed. We’ve used it to show that our architecture supports any predictive model to forecast traffic speed, which can be integrated with our modified routing algorithm and help the end user to plan future travels with some valuable estimates about the traffic delay.

6. PREDICTION OF FUTURE TRAFFIC

We have used an example to show how our system can be adjusted for route suggestions on a future date and time. To assess the route prediction capabilities of existing applications like Google Maps or Bing Maps, we compared their public route suggestions for a particular time on different days of a week, and found out that the suggested routes are the same. This does not qualify as a good predictive model as one would assume that the route suggestions should vary based on the predicted traffic on different days. We have addressed this problem using our algorithm wherein the suggested routes would vary on different days, given the predicted traffic speed is not the same. Our prediction is based on the traffic speed predicted by the support vector regression (SVR) model [9]. The use of SVR is not uncommon in traffic-level prediction and has been researched earlier [30]. This functionality would help the user to plan their travel in advance in a more realistic way.

For our experiments, we looked at some historical samples for traffic feed and synthetically generated our test data using a mean traffic-speed with an added variance in a normal distribution. This allowed us to have some level of confidence that our generated data was more realistic. We did traffic profiling on the generated test data-set to analyze the correlation between real-time traffic speed and the free flow (FF) speed. As expected, the ratio of the free flow speed to traffic speed would decrease during the peak hours of traffic as shown in Figure 5. For our prediction-modeling, we built a feature vector for each of the data points in our test dataset containing the historical traffic speed. The feature vector contains attributes like the Jam Factor (JF), Free Flow Speed (FF) and Route Information for which the real-time speed was recorded. The real-time values of these attributes are already available from the HERE Maps Traffic API as shown in Figure 4. The test values for these attributes were generated in a normal distribution curve, around a mean value with an added variance, similar to the way we generated our test data for historical traffic speed.

Once the feature vectors were generated for all data points, we used it to train a SVR [9] model to help us predict traffic speed on a future date and time. It is important to note that the choice of the model, parameter tuning and feature selection would play a major role in determining the accuracy of predicted traffic speed. However, we do not go into any such details as its beyond the scope of this paper.

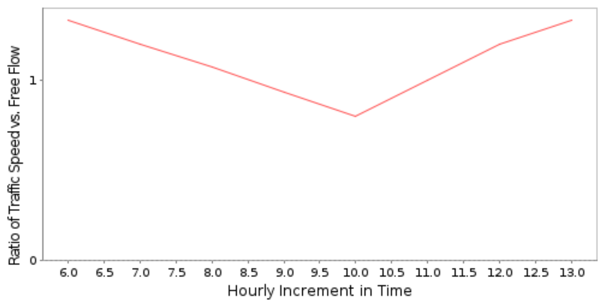


Figure 5: Traffic Speed Profiling

We conducted experiments to get route suggestions on a future date with our predicted traffic-speed and compare the results against the traffic predictions done by existing routing applications like Google Maps. The result for this experiment is presented in the next section.

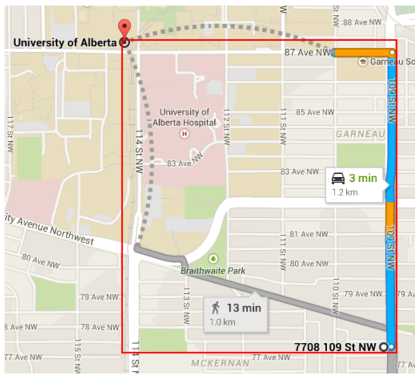


Figure 6: Bounding box between 7708, 109 St and University of Alberta

7. EXPERIMENTAL EVALUATION

In our experiment, we do the traffic analysis and route planning for different routes from Parkallen to University of Alberta in the city of Edmonton, Alberta, Canada. We choose the source location as 7708 109 Street NW and the destination location as University of Alberta, shown as the bounding box (red line) in Figure 6. The bounding box may have multiple routes from source to destination. We use our algorithm to analyze four different scenarios mentioned below.

1. Different walking speeds of a user results into different route suggestions for the same source and destination location at a fixed departure time.

2. Different level of real-time traffic might add some delay in the travel time, which could lead to different arrival time for same route-suggestion.

3. How a future route suggestion based on traffic speed prediction can differ from existing approaches.

4. Our system imposes minimal overhead on the existing route-suggestion techniques.

The first part of our experiment is done to show how different walking-speeds can change the routing suggestions. The Figure 7 sums up our result of this experiment. In this part, the source location is 7708, 109 Street, and the destination is University of Alberta. The distance between source and destination is nearly 2 km. This small distance is enough to show the difference of results with varied walking speed. We have conducted this experiment for three types of walking-speed: 0.5 m/s (solid line in Figure 7), 1 m/s (dashed-dotted line) and 1.5 m/s (dotted line). The starting time from the source location is fixed at 10:47:00 on 30th March'15 for this experiment. As it can be seen from the Figure 7, the leftmost trip (solid line) departs from the source location at 10:47:06 and reaches at 11:18:34. The trip at center (dashed-dotted line) departs at 10:48:53 and reaches at 11:07:52. They both go through the same bus-stop, but they follow different bus-numbers (4 and 94). We can also notice that the walking times for the first part of their journeys to the bus-stops are different, in spite of distance being same. The rightmost trip (dotted line) is different in multiple aspects from the other two. The walking distance is only 160 m for this trip, and the bus-stop is also different. This trip catches Bus number 313 and reaches before the other two trips at the destination.

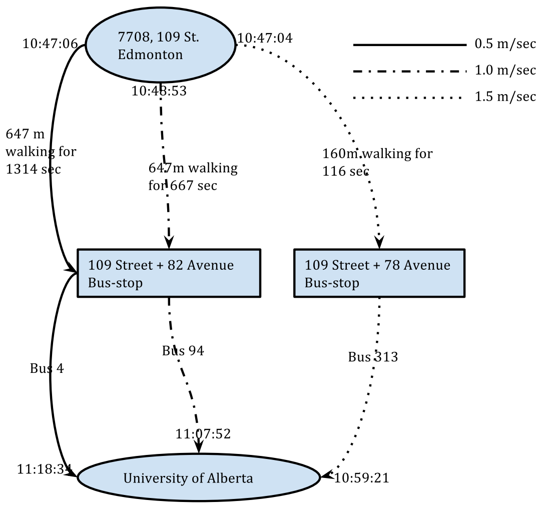


Figure 7: Routing results for different walking-speeds

This experiment shows how our system displays different types of routes based on the walking-speed. It should be noted that in our actual implementation, we take walking speed from accelerometric sensors (like Google Fit service). If we imagine that there are three different types of users who are simultaneously searching for same route at the same time on our system, they may get different results (like Figure 7) based on their commuting behavior, recorded by sensors. Our demonstration proves that this should be beneficial in terms of accuracy in route-planning.

Table 1: Free-flow (FF) vs. Real speed (RS) and their Arrival-time Comparison

FF vs. RS	Arrival Time
3.0	Mon Mar 30 11:12:42 MDT 2015
2.75	Mon Mar 30 11:11:52 MDT 2015
2.5	Mon Mar 30 11:11:03 MDT 2015
2.25	Mon Mar 30 11:10:13 MDT 2015
2.0	Mon Mar 30 11:09:24 MDT 2015
1.75	Mon Mar 30 11:08:34 MDT 2015
1.5	Mon Mar 30 11:07:44 MDT 2015
1.25	Mon Mar 30 11:06:55 MDT 2015
1.0	Mon Mar 30 11:06:05 MDT 2015

The next part of our experiment demonstrates how different traffic levels can affect the arrival time of a trip following the same route. Let us look at the route that we have shown previously in Figure 2 as reference. With that example, we assume that the traffic is only on the 82nd Avenue for our demonstration. So, the length of the road in which the traffic is simulated, is 726 m (324 + 200 + 202). We vary the ratio of traffic (FreeFlow Speed / Real Speed) from 3.0 to 1.0. The ratio 1.0 implies that there is no traffic, and the bus will be driven in its normal speed. The result we get with ratio 1.0, is the same as we get from applications such as HERE Maps or Google Maps. In Table 1, we can see that if the traffic level is as high as 3.0 in the first row, the trip is delayed to 11:12:42 from its normal (with ratio 1.0) 11:06:05. As the traffic ratio decreases, the arrival time at the destination also comes down. This experiment shows that our system is able to adapt to different traffic-levels and gives arrival time accordingly, which is not possible in state-of-the-art methodologies.

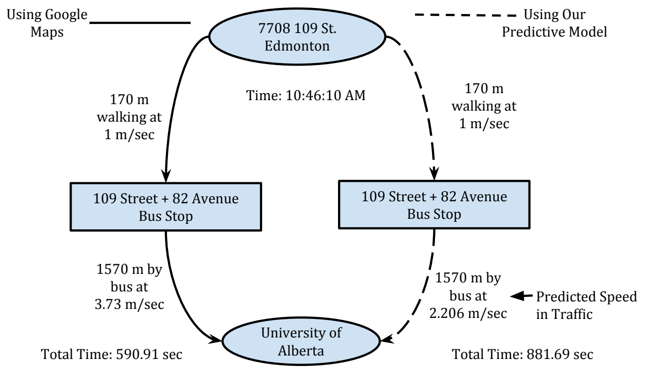

Using Google Maps
7708 109 St. Edmonton
Using Our Predictive Model
170 m walking at 1 m/sec
Time: 10:46:10 AM
170 m walking at 1 m/sec
109 Street + 82 Avenue Bus Stop
109 Street + 82 Avenue Bus Stop
1570 m by bus at 3.73 m/sec
1570 m by bus at 2.206 m/sec
Predicted Speed in Traffic
University of Alberta
Total Time: 590.91 sec
Total Time: 881.69 sec

Figure 8: Comparison of routes using predicted traffic speed

In the third part, we demonstrate the effectiveness of our system in predicting future route-suggestions. In Figure 8, we compared our predicted result with Google Maps static result. It is to be noted that this test was conducted on March 23, 2015 for traffic prediction on a future date: April 06, 2015. As shown in Figure 8, our model takes into account the predicted traffic speed as opposed to Google Maps which has the same bus speed on different days of the week for future route suggestions. Although this is done on synthetic data-sets described in Section 6, we can assume real-data set would also reflect similar kind of behavior with our system.

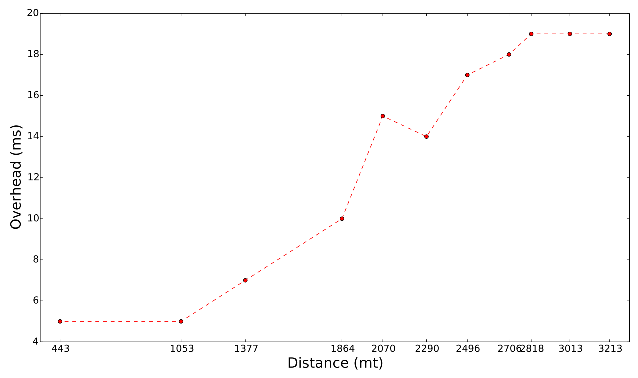


Figure 9: Overhead of our algorithm with respect to increasing distance

Lastly, we measure the overhead of our algorithm with increasing distance. We have chosen Route 51 of Edmonton Transit System [13]. We have conducted our experiment on a route familiar to us because of proper and reliable interpretation of poorly documented HERE Maps Traffic response. The Route 51 is from Parkallen, Edmonton, AB, Canada to University of Alberta, AB, Canada and is nearly 3 km long. The Figure 9 represents the overhead of our algorithm with increasing distance. We have taken all the bus-stops on the route and calculated overhead of our system for the distances from those bus-stops to the destination. We can see in Figure 9 that as the distance increases, the overhead also goes higher because traffic-speed needs to be calculated for more number of roads. The maximum overhead for a 3 km Route is as small as 20 ms. In this experiment, we have made an assumption that we already have the real-time traffic data available in our system. This is a fair assumption because generally a system should periodically store real-time traffic data in its memory, and traffic-data should be available on demand. Even if we assume the traffic-feed is stored in disk, we can use caching for faster results. This kind of caching is pretty common in case mapping services [25, 28] to improve performance and reduce response-time. Overall, we can see that our algorithm has fairly minimal overhead, provided that common caching technique is employed for maps.

8. CONCLUSION AND FUTURE WORK

This paper presents a novel architecture which aims to support varying traffic levels on the roads and commuting behavior of the end users for public transportation. To support the robustness of our architecture, we have developed a working system and conducted experiments to evaluate our results. The results prove that our architecture is supportive of our claimed objectives with an overall minimal overhead.

We have developed our system-prototype in Java [2], in some parts using RESTful architecture. Our prototype implementation can be modeled as standalone application on mobile-platforms (e.g - Android) or in web-services. We plan to show our suggested routing results in visual maps for better representation. On the other hand, there can be improvements on the historical profiling of traffic data. An efficient

[2]Available at https://github.com/sohamm17/MapsTraffic

predictive model for traffic levels can help the end users plan their journey in advance with good reliability in the system. We plan to use our system with historical data provided by PeMS system [1] of California State Government for traffic prediction as done in [4]. Moreover, other traffic incidents like road accidents, constructions, bad weather conditions etc. can be integrated with our routing algorithm to further improve the route suggestions. Lastly, studying the user-satisfaction of our system will certainly be very interesting, but needs planned deployment on the public domain.

9. ACKNOWLEDGEMENT

We would like to thank Dr. Mario Nascimento, Prof. and Chair of the Department of Computing Science, University of Alberta for his constant support and guidance throughout this project.

10. REFERENCES

[1] Caltrans Performance Measurement System (PeMS). http://pems.dot.ca.gov/.

[2] General Transit Feed Specification (GTFS). http://code.google.com/transit/spec/transit_feed_specification.html.

[3] HERE Maps. https://www.here.com/.

[4] Traffic Predict. http://trafficpredict.com/.

[5] Transit App. http://transitapp.com/.

[6] J. Aslam, S. Lim, X. Pan, and D. Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pages 141–154.

[7] H. Bast, P. Brosi, and S. Storandt. Real-time movement visualization of public transit data. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 331–340. ACM, 2014.

[8] M. Catala, S. Dowling, and D. Hayward. Expanding the google transit feed specification to support operations and planning. Technical report, 2011.

[9] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[10] S. Clark. Traffic prediction using multivariate nonparametric regression. *Journal of transportation engineering*, 129(2):161–168, 2003.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

[12] T. T. Cuong. Crowdroute: A crowd-sourced routing algorithm in public transit networks. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, GEOCROWD '13, pages 9–14.

[13] Edmonton Transit System. http://www.edmonton.ca/transportation/edmonton-transit-system-ets.aspx.

[14] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 794–805. VLDB Endowment.

[15] Google. Stuck in traffic? http://googleblog.blogspot.ca/2007/02/stuck-in-traffic.html.

[16] Google. The bright side of sitting in traffic: Crowdsourcing road congestion data. http://googleblog.blogspot.in/2009/08/bright-side-of-sitting-in-traffic.html.

[17] Google Inc. Google Fit. https://fit.google.com.

[18] Google Inc. Google Maps. http://maps.google.com.

[19] B. Kerner, C. Demir, R. Herrtwich, S. Klenov, H. Rehborn, M. Aleksic, and A. Haug. Traffic state detection with floating car data in road networks. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 44–49.

[20] F. Lécué, S. Tallevi-Diotallevi, J. Hayes, R. Tucker, V. Bicer, M. L. Sbodio, and P. Tommasi. Star-city: Semantic traffic analytics and reasoning for city. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI '14, pages 179–188.

[21] G. Leduc. Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*, 1:55, 2008.

[22] P. H. Li, M. L. Yiu, and K. Mouratidis. Historical traffic-tolerant paths in road networks. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 477–480, 2014.

[23] Microsoft. Bing Maps. https://www.bing.com/maps.

[24] W. Min and L. Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606 – 616, 2011.

[25] G. Pelosi and G. Psaila. Smac: Spatial map caching technique for mobile devices. In *Proceedings of the ACM Symposium on Applied Computing*, SAC '10, pages 1829–1830.

[26] C. Quek, M. Pasquier, and B. Lim. Pop-traffic: a novel fuzzy neural approach to road traffic analysis and prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):133–146, June 2006.

[27] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 85–98.

[28] M. R. Vieira, P. Bakalov, E. Hoel, and V. J. Tsotras. A spatial caching framework for map operations in geographical information systems. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 89–98. IEEE.

[29] Waze. https://www.waze.com/.

[30] C.-H. Wu, J.-M. Ho, and D.-T. Lee. Travel-time prediction with support vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):276–281, 2004.

[31] T. Zhou, L. Gao, and D. Ni. Road traffic prediction by incorporating online information. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 1235–1240, 2014.