Bus Congestion Data Source Report

Contributor:

Name	**Email address**
Chengjun Wu(Lead)	simonwu@bu.edu
Annan Miao	annan@bu.edu
Jing Li	jingli18@bu.edu

1. Overview

As described in our project proposal, we plan to build up a model to see how likelihood a certain place would be in heavy traffic and in dangerous condition through taking advantage of MBTA historical traffic data. By leveraging this model, we are able to figure out the traffic pattern in Boston and then find out how to avoid 'bad' routes for buses of Boston public school.
Starting as the first step for project, we've been researching the following data sets and find a few more actions need to be taken to get them clean for further programming.

2. Data Source

In this section, we will elaborate on how we clean and compile each data set. In addition to that, we will also illustrate the underlying reasons of the way we process for each.

- BPS Sensor Data

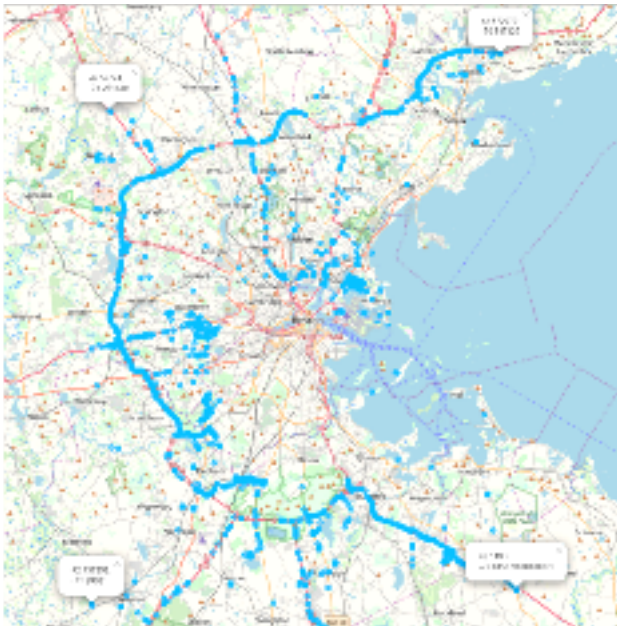
  This data set was given in a CSV file, which has the data frame shows below:

Logtime	**Latitude**	**Longitude**	**Heading**	**Speed**	**VendorhardwareID**
2018-02-07 00:23:48	42.33514	-71.0803	0	1	MS122

1. This dataset has the information that we need about the school buses. It has more than 60,000,000 rows so that we use chunks to deal with it. The core information that we want for each row is the timestamp and location for a vehicle, which is specified by the machine ID of that vehicle. So we filtered this dataset and got this data frame:

Logtime	Latitude	Longitude	VendorhardwareID
2018-02-07 00:23:48	42.33514	-71.0803	MS122

2. The target is the BPS routes, but this dataset is not sufficient to generate the model to analyse the traffic pattern of the Great Boston Area. So, we need MBTA dataset to be the source for our model. The MBTA has 168 routes and we want to filter these according to the BPS routes area which we think can help us get rid off the routes that we do not concern. Here are how we did this process:
    a. Get the locations of all vehicles in BPS dataset and set the chunk size to 10000 so that we dealt with the data chunk by chunk.
    b. In each chunk, we got four indexes for maxLatitude, maxLongitude, minLatitude, minLongitude and save this location points. There are many dirty points whose location is out of the US. we firstly analyse the MBTA service area to filtered the BPS dataset.
    c. We used folium to scatt these points on Map and got the boundary map for BPS dataset, which is shown below:



maxLongitude, maxLatitude, minLongitude, minLatitude = -70.845879, 42.572979, -71.29081, 42.137852

d. We used these four boundary points to filtered the MBTA routes. We traverse through all 168 routes to analyse whether this route should be ignored. For each routes, we check through all its stops and counted all stops which are outside of the boundary. Firstly we decided to delete all routes which has no intersection of the boundary area. But there is no such route. Then we checked the o*utsideStops : allStops* ratio, if this ratio is bigger than 0.9 then we deleted this route. By this way we got 8 routes that can be deleted, which is really small according to the whole routes. At the end we decided to keep all the 168 MBTA routes to analyse.

- MBTA Historical Traffic Data

Data Source: https://api-v3.mbta.com/docs/swagger/index.html

As we've been checking out from any web portals online, there's no such a web portal can provide us with the MBTA historical traffic data in form of real-time location. Thus, we decide to scrape these real-time location data for every vehicle(bus) in MBTA system for two weeks to construct a new traffic data set for training our model.
If we keep track of every bus by looking at its location every 3 minutes and save that location as well as some other useful attributes as an entry, the new traffic data set will include quite a lot of entries even though it's just 14 days in length. We think that amount of data is informative enough to give us a traffic pattern in Boston.

Here's the form of each entry in the output file and the descriptions for each attribute:

id	The identifier for different buses
current_status	whether this bus is in service or not
direction_id	The direction is bound to. Inbound or outbound.
latitude	real-time location
longitude	real-time location
updated_at	The timestamp
day	The day
time_slot	The hour of the timestamp that used to indicate what time interval it lies in. E.g The time_slot of 17:00:01 will be 17.

1. From MBTA APIs, we find there are 168 bus routes within MBTA system where almost all routes here may have impacts on the routes of Boston public school bus. For that case, we decide to keep them all to build model rather than filtering any out.
2. Our script has been deployed to auto-fetch the real-time information of all vehicles running on the 168 routes described above every 3 minutes from MBTA APIs. On average, there will be over 5000 entries written in the output file for an hour.
3. Each entry in the output file represent a piece of real-time information for a certain bus at a certain time. The first six attributes are included in the response from MBTA APIs while the latter two are variables defined and computed by our own for further researching.

● Boston Public School Location Data

The raw data was given in a website, which has the data structure shown below:

Adams, Samuel Elementary
Principal/Headmaster: Joanna McKeigue Cruz
165 Webster St.
East Boston, MA 02128
Grades Offered: K1-5
Hours: 7:30 a.m. - 2:10 p.m.
School Type: Traditional

There are 126 public schools in Boston area. The information we need is the school location to specify routes, and the hour of operations to detect traffic peak time. We use Google Map API to get the location represented by latitude and longitude from the name of public schools for later calculation. The cleared data has structure shown below:

School name	School hour	Location
Adams,Samuel Elementary	7:30 a.m. - 2:10 p.m.	(42.3695431, -71.0200613)

As stated above, we use BPS sensor data to achieve routes for school buses, and use MBTA historical data to analyze congestion situation of BPS routes. With the information provided, we can then use Google Map API to avoid congestion by selecting alternate routes between stops in congestion area.
The method direction of Google Map API would take the location of start and end points, and return the direction in json format. In this case we should set the transportation mode to “driving” to get alternate bus routes.