Traffic Data Flow Analytics: Waze vs Bluetooth

Brooklynn Hauck[1], Tracy Liu[2], Dr. Lee Han[3], Dr. Kwai Wong[3], Nima Hoseinzadeh[3], Yuandong Liu[3]

1. Slippery Rock University

2. Changsha University of Science and Technology

3. University of Tennessee

Abstract

This paper will discuss the necessary steps we had to take to achieve our goal as well as some of the issues we had to overcome. The goal of this project was to find out whether Waze was a reliable source of data to use as the ground truth for research purposes. This information could be useful for researchers who are already using or are considering using Waze as a ground truth. To achieve this goal we will be using Bluetooth as the ground truth to our research, as well as Comet through openDIEL, Rstudios, and scripting.

Introduction

Background

Waze is a downloadable GPS app, provided by Google, that users can download for both IOS and Andriod devices. Users can get turn-by-turn navigation as well as user-summited travel times and route details. Over millions of users have downloaded Waze and have been using it on their travels, allowing Waze to college millions of data on speed and travel time. Due to the amount of data provided by Waze on their online data source Universities have started to use the data as their ground truth for research. This would not be an issue if Waze data had been tested on reliability before being used for research but the data provided by Waze has not been tested for reliability.

Bluetooth is a wireless technology standard for exchanging data between fixed and mobile devices over short distances using short-wavelength radio waves. The range for Bluetooth connection is typically less than 10 m up to about 100 m. With newer versions of Bluetooth, the connection can be between 40 and 400 m. This means that when collecting data from Bluetooth, detectors will need to be put within these short distances from vehicles. To do this, most detectors are hooked up to red lights so that when vehicles drive underneath lights data will be collected. Bluetooth can also be considered a reliable resource since it has already been tested by Ali Haghani, Masound Hamedi, Kaveh Farokhi Sadabadi, Stanley Young, and Philip Tarnoff called Data Collection of Freeway Travel Time Ground Truth with Bluetooth Sensors. This

paper stated that the results of their research showed that the new technology of Bluetooth Sensors is "a promising method for collecting high-quality travel time data that can be used as ground truth for evaluating other sources of travel time and other intelligent transportation system applications"[2]

All data from both Bluetooth and Waze was first pulled from either the detectors or from the open-source. Open Waze data source captures data in real-time using a JSON file type while Bluetooth data is captured in XML files. To use the data both the JSON files and the XML files must be converted to the same file format. So they were converted to a CSV file format. Each CSV file contained one day's worth of data. This process was already completed for us when we were given the data needed for our research. We were given one month's worth of data of both Waze and Bluetooth data.

All codes that will be used during this research will be written in the coding language R. R is used for statistical computing and graphics. The libraries used in the R code are: Chron; ggplot2; interp; zoo; taRifx; forecast; imputeTS; dplyr; plotrix; reshape2; Metrics.

OpenDIEL is a software designed to allow many units of parallel code running seamlessly under a unified executable and to allow these individual programs to exchange data specified by the user[4] . We used the engine, under Comet, to parallelize the code so that we could complete our analysis with the large dataset and print out our graphs promptly.

Research Purpose

The purpose of the research is to test to see if Waze data is a reliable source of data to use it as the ground truth in research projects. We also want to test to see under what conditions the data from Waze is most reliable. We would like to answer these questions:

- What day of the week is the data from Waze most reliable?
- What Time of day is the data from Waze most reliable?
- Does Traffic Volume affect the reliability of the data from Waze?
- Does Speed after the reliability of the data from Waze?
- Does the length of the segment affect the reliability of Waze?

Steps

Data Processing

The first step towards answering the questions that we wanted to answers was to get all the data and sort through it. A code was written in Rstudios to sort through both Bluetooth and Waze data and do the following things:

- Pull Data based on the Date and Segment
- Removes Waze Historical Data
- Removes Duplicate Data
- Merge Bluetooth and Waze Data

Pulling Data

Before we could start processing the data we were given we needed to pull the data from the files that were given based on the date and the segment. The date and the segment number is read of a text file outside of the code and put into a variable to be used throughout the rest of the code. The code to read the date and the segment and put them in a variable is shown in figure 1.

```
#Read the Date and Segment
fileName <- "DateAndSegment.txt"
OpenFile <- file(fileName,open="r")
data <-readLines(OpenFile)
Date <- data[1]
Segment <- data[2]
```

Figure 1: Code to get the Date and Segment

We then use the variables containing the data and the segment to sort through the Bluetooth and Waze data. We only pull the data for a particular date and segment.

```
#Read the Data: "bl" is Bluetooth Data and "wa" is Waze Data
WazeDate <- format(as.Date(Date), "%Y%m%d")
WazeFileName<-paste("WAZE_20190102-20190202\\RT_",WazeDate,".csv", sep = "" )
waze<-read.csv(WazeFileName)

BluFileName<-paste("Bluetooth data\\", Date, ".csv", sep ="")
blu<-read.csv(BluFileName)

# define "ii" as the segment number (Highest is 31)
ii<-as.integer(Segment)

# subset waze and blutooth data for segment "ii"
bl<-subset(blu, as.character(blu$Pairing)== as.character(st$Name[ii]))
wa<-subset(waze, as.character(waze$From.Street)==as.character(st$FromWaze[ii])
           & as.character(waze$To.Street)==as.character(st$ToWaze[ii]))
```

Figure 2: Code to pull data based on the Date and Segment

Removing Waze Historical Data

While sorting through the Waze data we came across an issue that may not seem like an issue at first. We had discovered that if Waze does not produce any date for a certain time of day, it will use an average of data from previous days to fill in this empty spot. This is called historical data. While this may not seem to be an issue, it causes the data to be inaccurate less reliable. To fix this we wrote one line of code to sort through the Waze data and remove all historical data (**Figure 3**). The code looks for a time the historical speed equaled the speed value. After removing the historical data we saw a drastic change in the shapes of the graphs we printed out later on. Figure 4 shows what the graph looks like before removing the historical data while figure 5 shows the graph after removing the historical data.

```
#Remove Historical Data
wa<-subset(wa, as.character(wa$Historical.Speed..mph.)
           != as.character(wa$Speed..mph.))
```

Figure 3: Code to Remove Historical Data from Waze

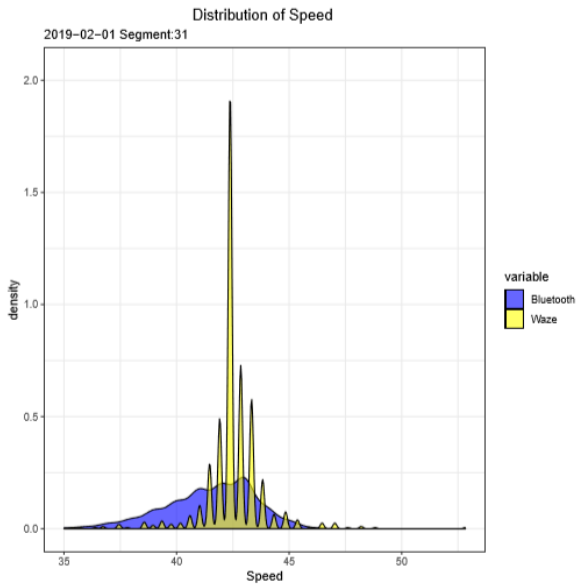


Figure 4: Graph with Historical Data

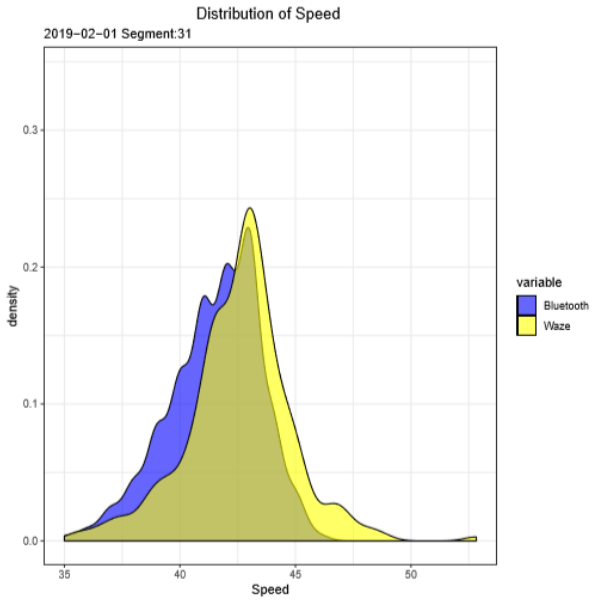


Figure 5: Graph without Historical Data

Removing Duplicate Data

While Historical data is an issue duplicate data is also an issue as well. Having duplicate data makes the data large and can make graphs harder to read as well as make the data less reliable. So we wrote two lines of code to remove the duplicate data from both Waze and Bluetooth. The code looks for any data that doesn't have a duplicate time stamp and pulls it into a variable removing all the duplicate data.

```
# Remove duplicate
bl<-bl[which(duplicated(bl$LastMatch)=="FALSE"),]
wa<-wa[which(duplicated(wa$Time.Stamp)=="FALSE"),]
```

Figure 6: Code to Remove Duplicate Data from Both Bluetooth and Waze data

Merging Bluetooth and Waze Data

To start comparing the data from both Waze and Bluetooth, we had to merge them. We merge the data from Bluetooth and Waze based on the time of day. Before merging the data we removed unnecessary columns and set the time to make merging the data easier.

```
# clean waze and bluetooth data and remove non important variables
bl<-bl[,-c(1:6,9:11,13)]
wa<-wa[,-c(1:3,5:7,9,11)]

# define the time for bl and wa
bl$LastMatch<-strptime(bl$LastMatch, format="%Y-%m-%d %H:%M:%S")
bl$LastMatch<-strftime(bl$LastMatch, format="%H:%M:%S")

wa$Time.Stamp<-strptime(as.character(wa$Time.Stamp), format="%Y%m%d%H%M%S")
wa$Time.Stamp<-strftime(wa$Time.Stamp, format="%H:%M:%S")
```

Figure 7: Code to remove columns and define time

After merging the data we change the column names and put them into a new variable to use later when calculating parameters and printing out graphs.

```
# Merge "bl" and "wa" with Time1
Mer<-merge(Time1,bl, by.x ="Time1", by.y ="LastMatch" , all.x =TRUE)
Mer<-merge(Mer,wa, by.x ="Time1", by.y ="Time.Stamp" , all.x =TRUE)

# colum name for final data
colnames(Mer)<-c("Time", "blTT", "blS", "waTT", "waS")

# put Merged file in df
df<-Mer
df$Time<-as.POSIXct(df$Time, origin = "1970-01-01", tz = "UTC", "%H:%M:%S")
```

Figure 8: Code to Merge Bluetooth and Waze Data

Calculate Parameters

To evaluate the difference of speed data between Waze and Bluetooth, the mean absolute error (MAE), the root mean standard error (RMSE), and the mean absolute percentage error (MAPE) were selected as the valuation indicators. Corresponding equations for the indicators are expressed as follows.

MAE (Mean Absolute Error):

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \overline{y_i}| \qquad (1)$$

RMSE (Root Mean Standard Error):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \overline{y_i})^2} \qquad (2)$$

MAPE (Mean Absolute Percentage Error):

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N} \left|\frac{y_i - \overline{y_i}}{y_i}\right| \qquad (3)$$

where y_i and $\overline{y_i}$ are the $\mathrm{i}-\mathrm{th}$ speeds in Bluetooth data and Waze data for interval i respectively. N is the number of Bluetooth or Waze speed samples in each hour.

As the data samples are not enough to get those indicators in each second, we aggregate them in 1 minute and calculate those indicators in each hour of each date, also the mean indicators through dates, the mean indicators of different segments are expressed as follows.

	MAE	RMSE	MAPE
Segment 1	2.1892	2.6288	0.0469
Segment 2	2.4704	2.9115	0.0515
Segment 3	2.2663	2.6334	0.0505
Segment 4	7.8430	8.7186	0.2052
Segment 5	6.0807	6.9048	0.1573

Segment 6	8.2280	8.9959	0.2684
Segment 7	4.2072	4.7272	0.1366
Segment 8	16.3164	16.5005	0.4805
Segment 9	6.0667	6.3625	0.1341
Segment 10	7.2523	8.1692	0.1924
Segment 11	7.6703	8.5503	0.2293
Segment 12	3.1369	3.6451	0.0907
Segment 13	3.0609	3.5602	0.0640
Segment 14	2.0104	2.3866	0.0429
Segment 15	1.9895	2.3519	0.0419
Segment 16	7.1739	8.0068	0.1721
Segment 17	2.8045	3.3861	0.0599
Segment 18	3.1959	3.6432	0.0766
Segment 19	4.4692	4.9151	0.1192
Segment 20	7.4335	8.3444	0.1743
Segment 21	7.3808	8.5761	0.3010
Segment 22	6.0856	6.7456	0.2412
Segment 23	5.7263	6.2203	0.1603
Segment 24	3.7685	4.2955	0.1057
Segment 25	9.6134	10.4328	0.3068
Segment 26	10.6924	11.7192	0.2537
Segment 27	2.7086	3.2314	0.0749
Segment 28	11.5111	12.2354	0.3003
Segment 29	3.1410	3.5524	0.0804
Segment 30	3.6801	3.9767	0.1214
Segment 31	1.6250	1.9510	0.0386

It can be seen from the table that the MAE and RMSE of most segments do not exceed 10, the MAPE does not exceed 0.1, and the error value of a small number of segments is high, which may be caused by other factors, such as the location, length of segment and traffic congestion or accident. These factors will be discussed below.

OpenDIEL

OpenDIEL was used to run multiple different versions of the code to print out the graphs discussed in the data visualization section. We choose to do this after calculating roughly how many graphs we will be printing out. Since we had 31 segments for 30 days we multiplied those together due to each graph being for one day and one segment. We calculated that we would need over 900 graphs per variable. Since openDIEL is a software designed to allow many units of parallel code to run at the same time. We were able to run multiple copies to get all the graphs we needed. We had originally chosen to use Bridges through openDIEL to run all of our code, but due to having issues with getting the libraries we needed for our code to work, we ended up using Comet instead which in the end worked perfectly. We used openDIEL to run three different codes.

- Create Folders

- Produce Graphs
- Separate Graphs

Creating Folders

The first code that ran in openDIEL was only run once. The code starts by reading the start date and the end date of all the data available from a text file (Figure 12). It then finds the files to be copied into the folder later and sets that into a variable. It will also put the date into a variable for the folder name (Figure 13). A while look and a for loop are both used to create the folders, using the date and the segment number as the name, and to move all the files needed into the folder. The last thing it does it puts a text file that contains the date and the segment number into the folder for use later (Figure 14).

Figure 12: Code reading the start and end date from a file

```
CWD<-paste("/home/bhauch/opendiel-core-2019/APPLICATIONS/TRAFFIC/data")
#setwd(CWD)
fileName <- "StartAndEndDate.txt"
OpenFile <- file(fileName,open="r")
data <-readLines(OpenFile)
Date <- data[1]
StartDate <- as.Date(data[1])
EndDate <- as.Date(data[2])
close(OpenFile)
```

```
#identify the current folders of files to be copied later
current.folderWAZE <- "/home/bhauch/opendiel-core-2019/APPLICATIONS/TRAFFIC/data/WAZE_20190102-20190202"
current.folderBLUE <- "/home/bhauch/opendiel-core-2019/APPLICATIONS/TRAFFIC/data/BluetoothData"

#Set which files to move
WazeDate <- format(as.Date(Date), "%Y%m%d")
WazeFileName<-paste("RT_",WazeDate,".csv", sep = "" )

BluFileName<-paste(Date, ".csv", sep ="")

#Format Date
DirectoryDate <- format(as.Date(Date), "%b%d")
```

Figure 13: Code to prepare for files to be copied

Figure 14: Code to create folders and copy files as well as make the text file

```
while (StartDate <= EndDate)
{
  for (val in Segment)
  {
    #Create the New Directory
    directoryName<-paste("TF",DirectoryDate,"-",ii, sep = "")
    dir.create(directoryName)
    NewDirectory<-paste("/home/bhauch/opendiel-core-2019/APPLICATIONS/TRAFFIC/data/",directoryName, sep = "")

    # identify the folder being copied to
    new.folder <- NewDirectory

    # copy the files to the new folder
    setwd("/home/bhauch/opendiel-core-2019/APPLICATIONS/TRAFFIC/data/WAZE_20190102-20190202")
    file.copy(WazeFileName, new.folder)
    setwd(current.folderBLUE)
    file.copy(BluFileName, new.folder)
    setwd(CWD)
    file.copy("stnima.csv", new.folder)


    #Set Working Directory to New Directory
    setwd(NewDirectory)

    #Make DateAndSegment.txt File
    sink("DateAndSegment.txt")
    cat(Date)
    cat("\n")
    cat(val)
    sink()

    #Next Segment
    ii<-ii+1

    #Set Working Directory Back to Home Directory
    setwd(CWD)
```

Producing Graphs

We used openDIEL to then run the code talked about in the data processing section and the data visualization section. We ran all the days in one section at once, meaning we only have run those groups of code 31 times. This made it easier to get all the graphs we needed without having to do them over 900 times.

Separate Graphs

After collecting all the graphs, we wanted to split them up based on the segment they were and the day of the week. This process took two different codes. The first code was written in the script and was only ran once. This code created a folder and then pull all the graphs, from folders they are in after being created, into the new folder (Figure 15).

```
mkdir PDFs
cd PDFs
mkdir Speed
mkdir TimeTraveled
cd ..
for dir in */
do
echo $dir
cd $dir
ls *.pdf
cp Speed*Segment_5.pdf ../PDFs/Speed/5#
cp Time*.pdf ../PDFs/TimeTraveled
cd ..
done
```

Figure 15: Code to Pull all Graphs into a new folder

The next step was done all by hand. All the graphs were then separated into specific folders based on their segment number. After this then another code written in R was used to separate the graphs in each segment folder into the days of the week. The folders for the days of the week were created by hand. The R code used a while loop and an if statement to sort through all the days and put them into their specific folder (Figure 16).

```
while (StartDate <= EndDate)
{
FileName<-paste("TimeTraveled_",StartDate,"_Segment_31.pdf",sep = "")
if ( StartDate == "2019-01-02" || StartDate == "2019-01-09" || StartDate == "2019-01-16"
     || StartDate == "2019-01-23" || StartDate == "2019-01-30")
{ file.copy(FileName, "Wed") }
else if ( StartDate == "2019-01-03" || StartDate == "2019-01-10" || StartDate == "2019-01-17"
          || StartDate == "2019-01-24" || StartDate == "2019-01-31")
{ file.copy(FileName, "Thurs") }
else if ( StartDate == "2019-01-04" || StartDate == "2019-01-11" || StartDate == "2019-01-18"
          || StartDate == "2019-01-25" || StartDate == "2019-02-01")
{ file.copy(FileName, "Fri") }
else if ( StartDate == "2019-01-05" || StartDate == "2019-01-12" || StartDate == "2019-01-19"
          || StartDate == "2019-01-26" || StartDate == "2019-02-02")
{ file.copy(FileName, "Sat") }
else if ( StartDate == "2019-01-06" || StartDate == "2019-01-13" || StartDate == "2019-01-20"
          || StartDate == "2019-01-27")
{ file.copy(FileName, "Sun") }
else if ( StartDate == "2019-01-07" || StartDate == "2019-01-14" || StartDate == "2019-01-21"
          || StartDate == "2019-01-28")
{ file.copy(FileName, "Mon") }
else if ( StartDate == "2019-01-08" || StartDate == "2019-01-15" || StartDate == "2019-01-22"
          || StartDate == "2019-01-29")
{ file.copy(FileName, "Tues") }
else
{ print("Error") }
```

Figure 16: Code into order to separate the graph based on the days of the week.

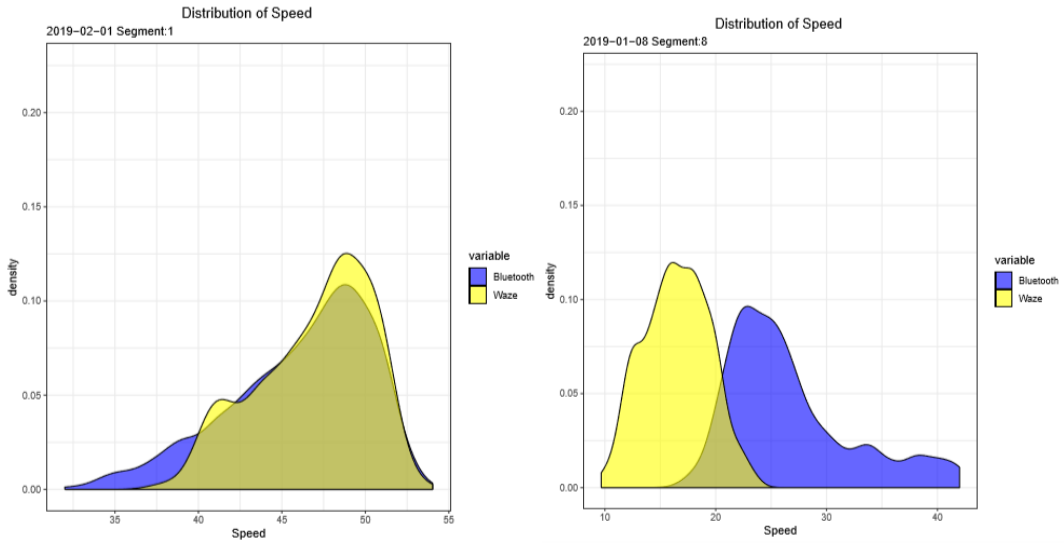
Data Visualization and Analyzation

Data Visualization

When printing out the graphs to compare the data, we decided to use four different types of graphs. We decided to use a distribution graph, a line graph, a boxplot, and a scatter plot. These graphs were used to compare four different variables, Speed, Time Traveled, Percent Error, and Traffic Volume. We choose the distribution plot to compare the speeds from both Bluetooth and Waze as well as the travel times. The line graph was picked to compare Percent Error and Traffic Volume. We choose the boxplot to show more data in the Percent Errors and compare the data in different speed groups. The scatter plots are used to show the different speeds that match between Bluetooth and Waze. Graphs are printed out with code written in R. The library used to print out graphs was ggplot2 and Metrics.

Data Analyzation

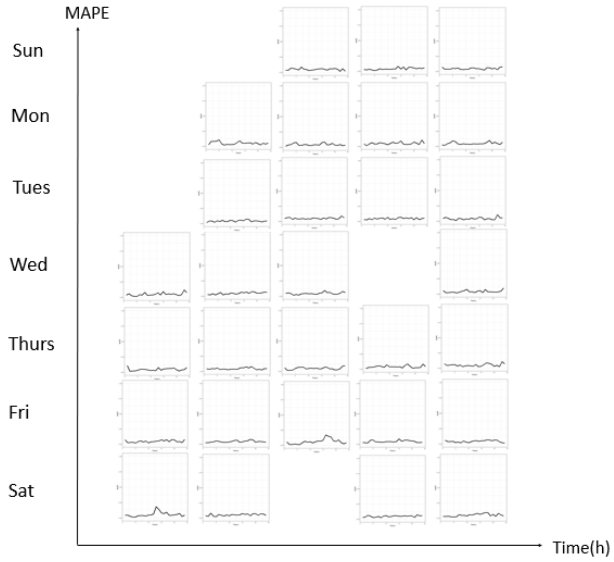
Distribution in Different Segments & Different Days

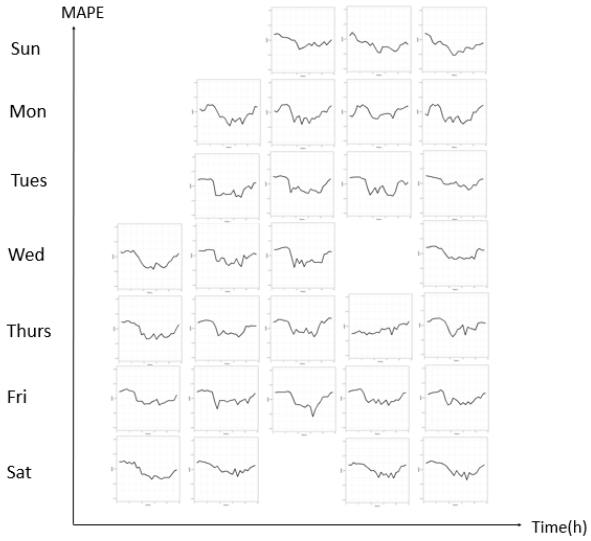


These graphs show the Distribution of the speeds between Waze and Bluetooth. Figure 1 shows a distribution that is similar between the two data set while Figure 2 shows a distribution that is not similar. It can be seen that when the distributions of the speeds between the two data set are similar, the speed is mainly distributed in the interval of 40mph to 50mph, and when the speed distribution is not similar, the distribution of the speed in Waze data tends to small, and the distribution of the speed in Bluetooth data is larger.

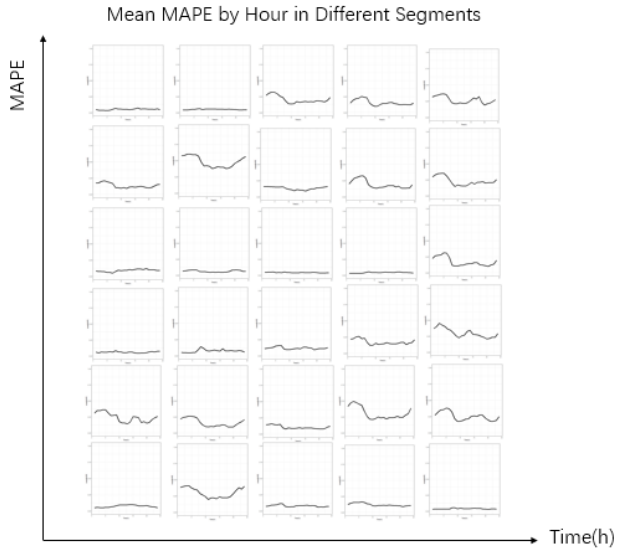
MAPE in Different Segments & Different Days

In the calculate parameters section, we have calculated the indicators hourly on each date of different segments in the two data set, including MAE, RMSE and MAPE. For better analysis, we make a line graph of MAPE for visualization, as shown in the figures below.

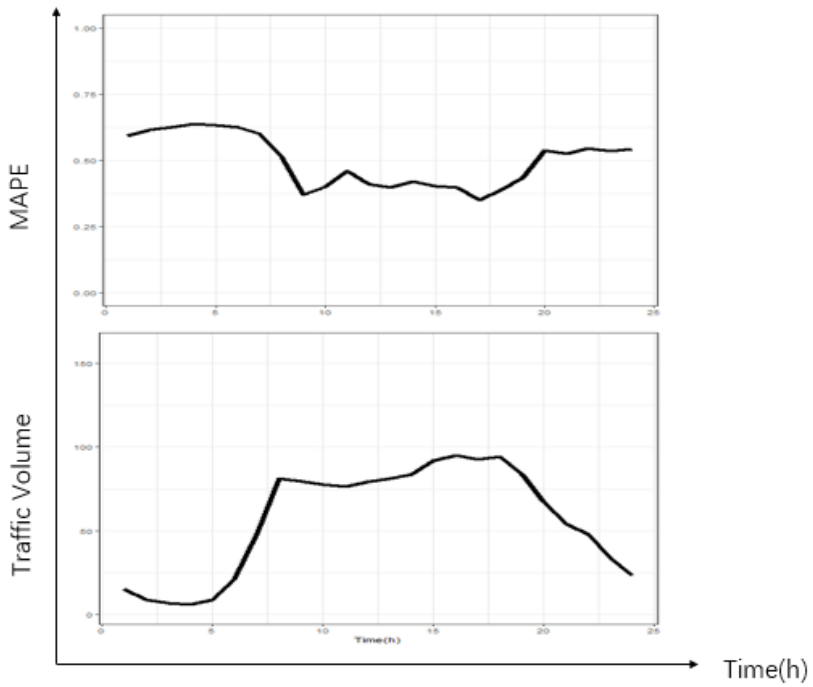




We have drawn the MAPE line graphs for each date of each segment, and combined by days of the week, we can see that the pattern of each segment is similar in different dates, which means there is no significant difference in the day at weeks, and the MAPE of different segments is very different. Figure 1: the MAPE line graph of segment 1, MAPE is very small and average. Figure 2: The MAPE line graph of segment 8, MAPE is large and the line patterns of different days are very uniform, low in the middle and high on both sides, like concave. During the daytime, MAPE is typically lower.



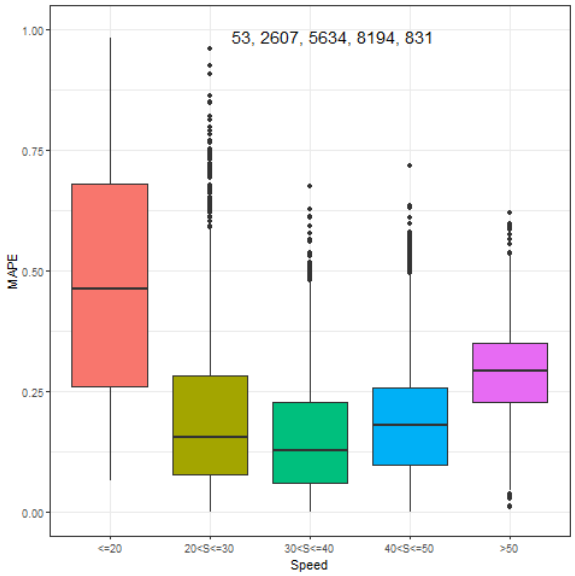
Then we calculate the mean MAPE through dates to get MAPE of different segments in 24 hours, and draw line graphs, combine them to compare the difference in different segments. The output is the same as the MAPE indicator we get, there is a big difference in different segments, and all the high MAPE segments' line graph has the same curve, which is low in the middle and high on both sides.



Considering that the traffic volume at the same time may affect the error indicator between the two data sets, we compare the traffic volume line graph on the same day with the MAPE line graph. The results show that the traffic volume is inversely related to MAPE. The larger the traffic, the smaller the MAPE.

MAPE in Different Speed Group

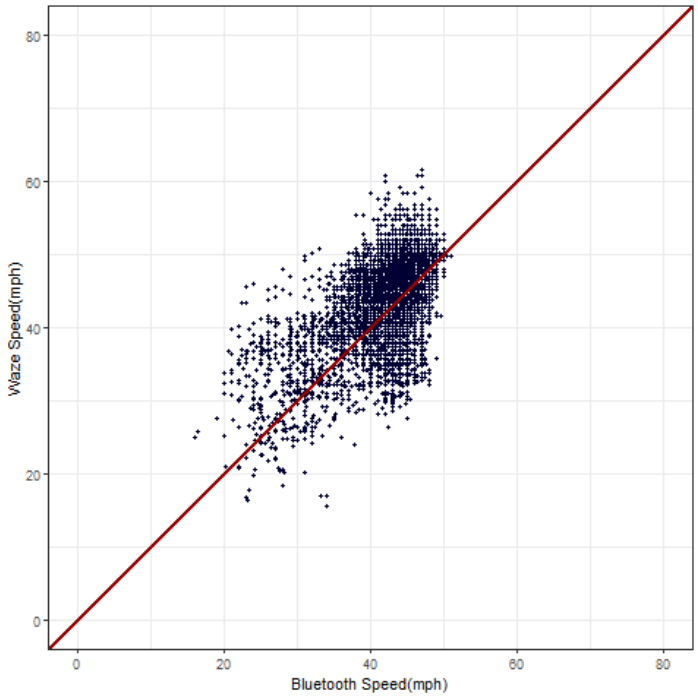
We observe the effect of speed on the MAPE between the two datasets through the box plot, and group the speed samples in the two datasets according to different speed levels, which are divided into less than or equal to 20, 20 to 30, 30 to 40, 40 to 50, and greater than 50 speed groups, and draw a box plot showing the value and distribution of MAPE.



It is obvious that when speed less than or equal to 20, or more than 50 mph, MAPE is higher than other speed levels. When speed is from 30 to 40 mph, MAPE is very low, which means Waze data is more reliable in this speed group.

Scatter

We show the speed of the two datasets at the same time as scatter plots, the abscissa is the speed of the dataset in Waze data, and the ordinate is the speed of the dataset in Bluetooth. We draw all the points through one month's data of each segment, in most segments, the scatter plot is close to the perfect line (Bluetooth speed = Waze speed).



Conclusion

We can say the Waze data is reliable enough to research with as long as it's under these conditions. Waze data can be from any day of the week since there is no significant difference between them. It must be during the daytime, not the night time due to there not being many vehicles driving at night. The higher the traffic the volume the lower the MAPE. The data must have a speed between 30 mph and 45 mph since we found that the majority of the data lies between those speeds. The data used should be from segments that are longer and shouldn't go through very many stoplights due to MAPE being found to be lower in those conditions while MAPE was found to be higher when the length was shorter and in lengths with more stoplights.

Acknowledgments

This project has been done during the tenure of the RECSEM program of 2018. The program took place at The University of Tennessee. The program is funded by JICS, NSF, and UTK.

Nothing could have been accomplished without the guidance of our mentors, Dr. Lee Han, Dr. Kwai Wong, as well as the graduate students, Nima Hoseinzadeh, Yuandong Liu.

References

[1] Ding, F., Chen, X., He, S., Shou, G., Zhang, Z., & Zhou, Y. (2019). Evaluations of Wi-Fi Signal Based System for Freeway Traffic States Monitoring: An Exploratory Field Test.

[2] Haghani, A., Hamedi, H., Sadabadi, K. F., Yound, S., & Tarnoff, P. (n.d.). Data Collection of Freeway Travel Time Ground Truth with Bluetooth Sensors.

[3] Yang, S., Brakewood, C., Nicolas, V., & Sion, J. (2019). Bikeshare Users on Budget? Trip Chaining Analysis of Bikeshare User Groups in Chicago. doi:10.1177/0361198119838261

[4] Wong, Kwai, et al., "Distributive Interoperable Exectuive Library (DIEL) for Systems of MultiPhysics Simulation." Accessed 31 July 2018.