Flyway: Predicting Foot Traffic in Open Spaces on Campus

ENGR-UH 4560 Machine Learning, Fall 2019

Nishant Aswani, *nsa325@nyu.edu* Barkin Simsek, *bs3528@nyu.edu*

I. Introduction

A. Motivation

Predicting traffic is one of the most common timeseries forecasting problems available for tackling. When placed in the context of smart cities, the problem is motivated by the question of ”how to enable users to make smarter choices when using transportation networks” [1]. However, this can be broadened to include how smart choices are made for using infrastructure and public spaces in general.

Given that our campus has multiple open spaces for students to study and/or relax, we want to be able to forecast the foot traffic that occurs in the multiple spaces. This could eventually be furthered, and the information organized, to provide a live prediction visualization of foot traffic around campus.

B. Problem Statement

Gathering from our previous experience with the New York City (NYC) taxi data set, we had a rough idea on how to approach timeseries data, along with the models and analysis that is carried out on such data. We were able to draw parallels between foot traffic in our project to pickup counts in certain locations in NYC.

In order to collect foot traffic, we determined that registering the device addresses would be a robust method involving little overhead. Students almost always carry their laptops and mobile devices with them as they move to study areas. Hence, our project could potentially focus on two topics: (1) how do we map the MAC addresses to the number of people and (2) are we able to forecast the ”busyness” of a given space. As tackling both of these aspects may not fit within our timeframe, we have decided to focus on the latter. We recognize the former will remain a challenge; however, we will approach it either with some simplifying assumptions or a pre-existing solution.

II. Existing Body of Work

A. Past Projects

One of the inspirations of this project was the Waitz application developed by students at University of California San Diego (UCSD) [2]. They used Raspberry Pi’s, loaded with network monitoring firmware and placed at various locations around their campus, to approximate the number of students within a given area. They did so by logging the Bluetooth and MAC addresses of devices within range of the network sniffing Raspberry Pi’s. It is not clear how they tackled the issue of mapping the number of devices to the number of students.

However, their application is relatively simple as they did not use this data to forecast the ”busyness” of locations or make predictions. Instead, it seems that they focused more on data collection and visualization. As a result, we thought it would be interesting to implement a similar tool for our campus; in the process, we hoped discover if there are any timeseries patterns we can exploit to provide useful information for students.

B. Literature

Unsurprisingly, there have been several attempts at using machine learning to determine ”crowdedness” or ”busyness”. One of the benefits of searching

through published work was being able to pick up on vocabulary within the field of research, making it easier to find other works.

We found that published work refers to this topic as "occupancy prediction" using "Wifi probing"[3]. Specifically, the work published by Wang et. al. determined five main characteristics in their occupancy prediction project. They realized their collected data had a "time-series characteristics" [3], validating our hypothesis that occupancy is relative to time of day, as well as the idea that past data can feasibly predict future data. While this may be trivial to the overall project, it was a source of confidence for our project as it implied we could successfully build a reasonably performing model.

The researchers implemented a "Markov-based feedback Recurrent Neural Networks (M-FRNN)" [3] model as part of their project to predict occupancy. They described the most basic neural network as having 3 layers, with the input layer accepting a vector of hashed MAC addresses, at a given time t. Moreover, they acknowledged that their raw data had fluctuations and noise that could potentially "deteriorate the prediction accuracy" [3]. To avoid this, they added a feature layer between the input and hidden layer, which calculates the "transfer probabilities" for every given MAC address at time t[3]. They describes their matrix of transfer probabilities (TPMI) as:

$$TPMI = \begin{bmatrix} X_k^{i-o} & X_k^{i-i} \\ X_k^{o-o} & X_k^{o-i} \end{bmatrix} \tag{1}$$

where each element is a conditional probability. For example, X_k^{i-o} refers to the conditional probability

$$X_k^{i-o} = \frac{\sum \text{from in to out}}{\sum \text{from in to in} + \sum \text{from in to out}} \tag{2}$$

They then use the quantities X_k^{o-i} and X_k^{i-i} as an updated input for their hidden layer. The output of the hidden layer is sent to something called the Context layer (serving as a short-term memory), which then feeds back into the hidden layer at each run of the network [3]. Their error function was simply a square of the actual occupancy subtracted from the predicted occupancy[3].
Another extremely important takeaway from this paper was the model assessment. Previously, we were only familiar with mean absolute percentage error (MAPE) and mean absolute error (MAE) as metrics. However, the researchers pointed out that these metrics "assume that an accurate prediction should have exactly same number as the observed occupants number" [3]. This is usually harmful to model assessment because a small tolerance should be granted. As long as the difference is minute, an "incorrect" output value can still be classified as an accurate prediction. Hence, they used a metric called the X-accuracy. They found that their M-FRNN model performed very well when the tolerenace (X) was set to 3, providing 80% accuracy on just the 3rd day.

Studying this paper was very crucial for us, as it gave us a starting point and some needed insight in how to approach the model formation. It was also encouraging to see that an example of a how typical machine learning model (ANN in this case) could be modified to account for the potential problems found in the data or problems inherent to the project itself.

III. Methodology

The methodology for realizing this project has 4 main steps: data collection, data processing, data analysis, and user interface. All of these steps are feasible by using the correct hardware with correct machine learning tools. In this section, the feasibility and applicability of the solution to the problem will be discussed by explaining the steps on the project in detail.

A. Data Collection

Data collection is the lowest level in the process and it is achieved by using ARM based Raspberry

Pi Zero computers since they have builtin bluetooth and WIFI chips. Besides having the WIFI chip as an hardware, the WIFI chip itself should be supporting the ”monitoring” mode for monitoring the wireless traffic. The WIFI chip of Raspberry Pi Zero also has this monitoring feature that enables the data collection for the project.

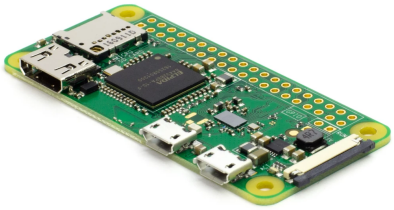

Fig. 1: Raspberry Pi Zero used for data collection

The Raspberry Pi Zero computer is running a special version of the Linux operating system and this also helps us with the data collection since Linux natively supports the `dumpcap` tool, which was used to collect the data packages in the air. Further details related to the `dumpcap` network traffic dump tool can be found here. Dumpcap is also capable of recording in both active and passive scanning modes that we can collect an extensive amount of data about the devices in the surrounding space.

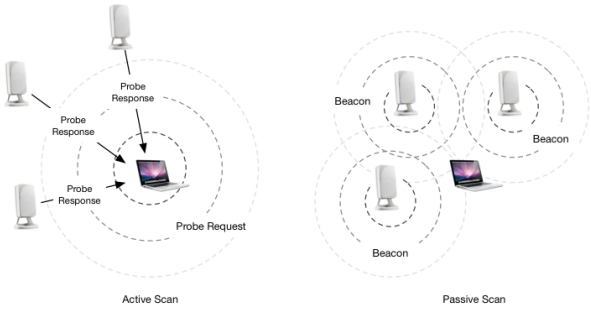

Fig. 2: The difference between the active and passive scanning is demonstrated [4]

When a WIFI chip is in the monitoring mode it is not capable of collecting private data since WIFI traffic between devices and the routers are already encrypted with some sort of password protection. That being said, we can still get various data regarding the source and destination of the data packages. These non-private information is enough for our project since we want to be able to approximately count the number devices and people in a space rather than spying on personal data. The following are the data fields that can be collected while monitoring WIFI signals in the monitoring mode:

- `frame.number`
- `frame.time`
- `wlan.addr`
- `wlan.ta`
- `wlan.ra`
- `wlan.sa`
- `wlan.da`
- `wlan.bssid`
- `wlan.fc.type`
- `wlan.fc.type_subtype`
- `radiotap.channel.freq`
- `radiotap.datarate`
- `radiotap.dbm_antsignal`

Here, `frame.number` and `frame.time` are the fields that contain date time information about the data package. These fields are useful for creating a time series data set out of the collected network traffic. The fields that start with `wlan` have the information regarding the source and destination of the data packets. This is information is given in the form of Media Access Control (MAC) addresses per each packet. This information is useful for counting and classifying the number of devices nearby to measure the crowdedness because MAC addreses are unique to every device and there are no two different WIFI devices with the same MAC address. Finally, the fields that start with `radiotap` have information regarding the signal strength, wireless frequency, and the data rate that channel support. Again, these fields are useful for understanding the proximity of the WIFI device and making decisions about whether to include or not to include certain devices into calculations.

For demonstrating the functionality of the system, more than 1 million data packages were collected over a week. The preview of the pandas dataframe that holds the data shows the collected data in Figure 3.

	frame.number	frame.time	wlan.addr	wlan.ta	wlan.ra	wlan.sa	wlan.da	wlan.bssid	wlan.fc.type	wlan.fc.type_subtype	radiotap.channel.freq	radiotap.datarate	radiotap.dbm_antsignal
0	1	17:44:56.905160884 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-45.0
1	2	17:44:57.090171784 +04	00:1b:b1:ab:7b:af	34:97:f6:ac:9c:10	00:1b:b1:ab:7b:af	34:97:f6:ac:9c:10	00:1b:b1:ab:7b:af	34:97:f6:ac:9c:10	2.0	44.0	2457.0	1.0	-47.0
2	3	17:44:57.824466813 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-46.0
3	4	17:44:59.195412588 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-46.0
4	5	17:45:00.179597000 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-46.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1102857	1643	16:44:53.474302402 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-45.0
1102858	1644	16:44:54.738280707 +04	18:81:0e:6a:68:e0	34:97:f6:ac:9c:10	18:81:0e:6a:68:e0	34:97:f6:ac:9c:10	18:81:0e:6a:68:e0	34:97:f6:ac:9c:10	0.0	5.0	2457.0	1.0	-46.0
1102859	1645	16:44:54.809035503 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	fc:0a:81:45:19:df	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	2.0	32.0	2457.0	1.0	-46.0
1102860	1646	16:44:55.624824410 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-44.0
1102861	1647	16:44:55.727162632 +04	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	ff:ff:ff:ff:ff:ff	34:97:f6:ac:9c:10	0.0	8.0	2457.0	1.0	-46.0

Fig. 3: Preview of the pandas dataframe created with the collected network traffic

B. Data Processing

After the data collection, the data can be processed to remove invalid and empty values as usual. Later, the MAC addresses collected under the `wlan` fields should be validated using the MAC address rules defined by the Institute of Electrical and Electronics Engineers (IEEE). Later, the MAC addresses should be converted into their decimal representations. Normally, MAC addresses are concatenated hexadecimal values and they are stored as strings since they contain letters inside them. However, we cannot use strings with machine learning algorithms and we need to get the decimal representation of each mac address to use the machine learning algorithms for analysing data in the next step. In addition to this, storing MAC addresses of devices might violate privacy of individuals. Therefore, MAC addresses can be hashed using hashing algorithms like MD5 to anonymized the data collected. Hash algorithms generate a hexadecimal value based on the input. Therefore, using a hashing algorithm is also useful for converting the non-numerical MAC address data into a numerical value as shown in the example below:

```
import hashlib
mac = "34:97:f6:ac:9c:10"
obj = hashlib.md5(mac.encode())
hash_hex = obj.hexdigest()
hash_dec = int(hash_hex, 16)
print(hash_dec)
```

The output of the operation above would be 239801908384185789790176606167296229703 and it is not reversible, which means that the original MAC address is no longer stored. That being said the hashing function will generate the same number if same exact MAC address is given and this is helpful for finding and avoiding duplicates in the data while preserving the privacy. Thus, the data processing step is crucial for having high quality data for training.

C. Data Analysis

Data analysis is the most critical part of the project after collecting and processing the data. The first challenge that needs to be addressed is the identify people who just pass by the location or temporarily visit the location that is being monitored. This is important, otherwise there will be too much variation in the prediction and these predictions will not accurately represent the true crowdedness of the location being monitored. Based on the background research we have conducted, we realized that we can use Markov-based feedback Recurrent Neural Networks (M-FRNN) to overcome the problem identifying people who are just passing by the location. The details of the model are explained under the ”literature” subsection. Furthermore, we are going to use the signal strength and frequency information (data fields that start with `radiotap`) to filter signals that are coming from far locations that are outside of the monitored location.

Additionally, we know about the maximum capacity of each location based on the architectural plans and we can use that number and the our predicted number to have a sense about crowdedness of the location that is being monitored. Furthermore, the Institute of Electrical and Electronics Engineers (IEEE) assigns specific MAC address prefixes to specific companies and they publish this data publicly. So, we can actually differentiate different devices based on their brands and understand if a device is a laptop or a mobile phone or a tablet or a watch. By using this information, we are going to

fine tune the crowdedness prediction we generated by accounting the possibility of one person carrying multiple WIFI devices.

D. User Interface

After the data collection, processing, and analysis steps are completed, the final step is having a very basic user interface for presenting the predicted data. For achieving this, we are going to use Bootstrap library for designing the user interface itself and use the PHP language for making the webpage dynamic that it can update the data shown with the real time data. A SQLite database will be used to store the collected and processed data. We already have a web hosting and a web domain (nyuad.app) that we already have for other projects as well. So, the app is going to be located at nyuad.app/flyway. The progress of the app can be also seen at nyuad.app/flyway as well.

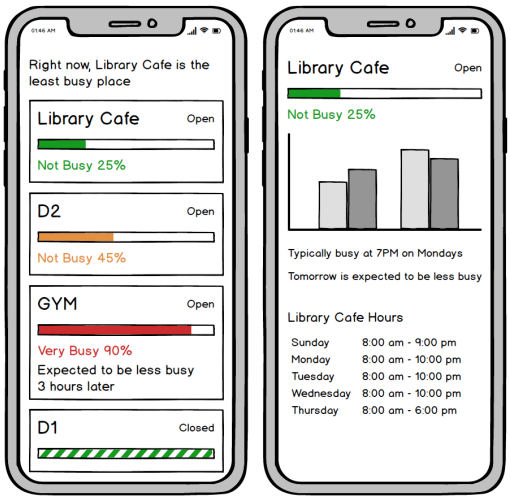

Fig. 4: Proposed user interface

IV. Conclusion

We have already figured out the exact tools and software that we need to successfully undertake this project. We also have some previous experience with these software and tools from previous projects. Given that this topic is well-covered, in casual materials as well as in peer-reviewed literature, we also believe there is enough inspiration and guidance available to us. Therefore, we believe that this project is feasible to build by using the tools and knowledge available to us.

References

[1] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[2] (Nov. 13, 2019). Waitz: Know before you go, [Online]. Available: http://waitz.io.

[3] W. Wang, J. Chen, T. Hong, and N. Zhu, "Occupancy prediction through markov based feedback recurrent neural network (m-frnn) algorithm with wifi probe technology," *Building and Environment*, vol. 138, pp. 160–170, 2018.

[4] (Nov. 13, 2019). Understanding the scan modes in wifi explorer pro, [Online]. Available: https://www.adriangranados.com/blog/understanding-scan-modes-wifiexplorerpro.