CS 498 Final Project - Kaggle Competition

Indoor Location and Navigation

Siddhant Sharma

ABSTRACT

This paper develops a Linear Regression model to predict the indoor position of smartphones based on real-time sensor data. The data-set contains metadata, training set and test set. The metadata contains basic details of the smartphone of the user in a mall and the floor details. The training set and the test set contain readings for Accelerometer, Gyroscope, Bluetooth Beacon, WIFI, Magnetic field, and Rotation vector of mobile devices in some Chinese malls. With the restriction of using only Bluetooth and WIFI readings, this paper describes a model to predict the location of a user in a Chinese mall using Linear Regression. The results are promising with low mean squared errors.

1 PROBLEM MOTIVATION

Localization and navigation has been a field of intense research activities because it enables a wide array of applications depending on response time and accuracy. Some of its uses are found in Location Based Services like Google Maps and location trackers like Snapchat and Instagram, Surveillance as such used by CCTV and Telemanipulation like the robotic arms used in the health industry.

"The Internet of Things (IoT) has emerged with the mission of enabling systems of communicating and interacting components with the capability of remote monitoring and control. These systems are designed to smoothly integrate into their surroundings within the context of many applications in urban and environmental monitoring and control, healthcare, industry, and military. The bold vision of the Smart City is built around such IoT systems with billions or even trillions of interconnected devices and sensors. The positive impact of such systems on various domains like infrastructure services, environment, and public safety have been demonstrated in numerous case studies"

As mentioned by **scientific** reports in *nature.com*, the importance of a proper location and navigation system is amplified by the emergence of the Internet of Things.

Wireless sensor network (WSN) is a network for detecting data and organizing it at a central location. It records physical phenomenon like pressure, humidity, temperature, etc., with the use of spatially distributed sensors which have the ability to communicate among themselves. Hence, for its localization process, the application of a proper localizing system is an absolute necessity. Thus, we can again see the importance of a good localization and navigation system.

The growing importance of good localization models shows why tackling the problem with the given data-set is a chance to figure something authentic out. With this data-set asking to building a model for indoor location and navigation in malls, I believe the localization issue would be the biggest tackle to face. The prediction of the axes values of the smartphones is the system around which the model needs to be built, and in the process, I would be able to play around and try different localization techniques that could lead me to something groundbreaking.

2 RELATED WORK

The importance of localization and navigation systems have become so important in today's world that several research papers have been published tackling different aspects of the problem.

One of the papers was "Review of indoor localization techniques". It starts by talking about the old navigation systems like location through stars and compares it to the modern technological advancements that have improved navigation systems, namely GPS. The paper states its problem statement as the review of indoor localization techniques as the solutions to indoor navigation is not as profound as outdoor. It gives an overview of the indoor localization and the basic structure of getting the location from the signals. Several signal measurement techniques are mentioned: Received signal strength (RSS) method, Time based method (Time of arrival, Time difference of arrival, Round trip time), Angle of arrival methods. It also mentions a couple positioning calculation techniques like Trilateration and Triangulation. The entire paper served as a review of indoor positioning techniques to further research in this field.

Another paper that caught my attention was "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications". The paper talks about the use of indoor localization techniques in modern communication and elaborates on the fundamental limitations of localization in indoor environments. It is followed by a detailed explanation of some of the basic positioning techniques in indoor environments. A lot of details are provided about the system-based localization in indoor environments like WLAN, Ultra-Wideband, Sensors, etc. and the challenges that follows are also mentioned. The paper is a long read but it perfect summarizes the basics of indoor localization and the advances in this field.

3 SYSTEM DESIGN

3.1 Reading the Dataset

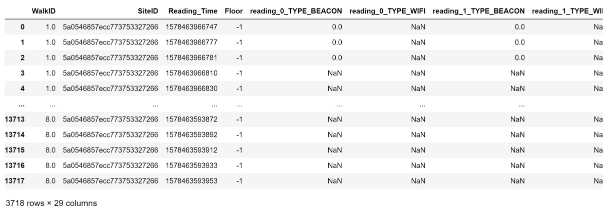
The data-set was downloaded from the Kaggle Competition website. It contained real-time sensor data like Gyroscope and WIFI readings for smartphones in Chinese malls. The data-set contained metadata, train set and test set. The metadata contained details about the smartphone and the floor plan. The train set and the test set contained readings from Accelerometer, Gyroscope, Bluetooth Beacon, WIFI, Magnetic field, and Rotation vector from a smartphone.

First, the metadata had a total number of 981 readings for Floor Images, Floor Info and Geo Maps. A script in Python was written to extract the metadata in the train files which were preceded with a # and had details about the smartphone like type, name, version, vendor, resolution, power, and maximum range. Another Python script was written to extract the readings from the training files. The readings were kept in a tabular format with the labels: WalkID, SiteID, Floor, Time, Type, reading_ $<(0-8)>$ _. The different types of readings were – *TYPE_ACCELEROMETER*, *TYPE_ACCELEROME–TER_UNCALIBRATED*, *TYPE_BEACON*, *TYPE_GYRO–SCOPE*, *TYPE_GYROSCOPE_UNCALIBRATED*, *TYPE_–MAGNETIC_FIELD*, *TYPE_MAGNETIC_FIELD_UNCA–LIBRATED*, *TYPE_ROTATION_VECTOR*, *TYPE_WIFI*.

The readings were a mix of floating numbers and characters, which was why cleaning of the data was necessary. Another reason for cleaning was that only WIFI and Bluetooth Beacon readings were allowed. Hence, most of the readings had to nullified. One of the most difficult aspects of reading the data was understanding the given files in the data-set and coming up with Python scripts to extract the data in a concise table for easier calculations later. Moreover, we had to read the sample submission .csv file to have an understanding of how the solution output should write. A majority of the time was spent on reading the dataset to come up with a viable solution to the problem in hand.

3.2 Cleaning the Dataset

The downloaded data-set had an incredible amount of data. The data had to be cleaned to extract out the necessary elements that would be required to build the model. First part of cleaning data came with separating out the waypoint details into a data frame called *waypoint_df*. It contained readings with the following labels – WalkID, SiteID, Time, Floor, x, y. Of these, the first 4 were metadata and the x and y readings were taken from *TYPE_WAYPOINT* from the train data. *waypoint_df* is displayed to make sure the extraction was successful. This extraction would later come in handy during the extrapolation of data.



	WalkID	SiteID	Reading_Time	Floor	reading_0_TYPE_BEACON	reading_0_TYPE_WIFI	reading_1_TYPE_BEACON	reading_1_TYPE_WI
0	1.0	5a0546857ecc773753327266	1578463966747	-1	0.0	NaN	0.0	Na
1	1.0	5a0546857ecc773753327266	1578463966777	-1	0.0	NaN	0.0	Na
2	1.0	5a0546857ecc773753327266	1578463966781	-1	0.0	NaN	0.0	Na
3	1.0	5a0546857ecc773753327266	1578463966810	-1	NaN	NaN	NaN	Na
4	1.0	5a0546857ecc773753327266	1578463966830	-1	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	
13713	8.0	5a0546857ecc773753327266	1578463593872	-1	NaN	NaN	NaN	Na
13714	8.0	5a0546857ecc773753327266	1578463593892	-1	NaN	NaN	NaN	Na
13715	8.0	5a0546857ecc773753327266	1578463593912	-1	NaN	NaN	NaN	Na
13716	8.0	5a0546857ecc773753327266	1578463593933	-1	NaN	NaN	NaN	Na
13717	8.0	5a0546857ecc773753327266	1578463593953	-1	NaN	NaN	NaN	Na

3718 rows × 29 columns

Figure 1: train_df_sparce

The next step was to extract and clean the remaining data from the train set that contained the sensor readings. The sensor data was to be converted into decimal floats. Before doing that, certain columns had to be converted for *TYPE_WIFI* and *TYPE_BEACON* into NaN's as they contained text strings in them. After the conversion, the data was extracted into in a table with the labels: WalkID, SiteID, Time, Floor, reading_ $<(0-8)>$ _ $<TYPE>$. Next, the columns that were restricted were dropped and only the readings for WIFI and Bluetooth Beacon were kept. This table was stored in *train_df_sparce*.

The most important part in the cleaning was the separation of *waypoint_df* and *train_df_sparce* as it would later be required during interpolation. With the unnecessary data removed, the data-frame became a lot easier to read and understand which would help in building the model.

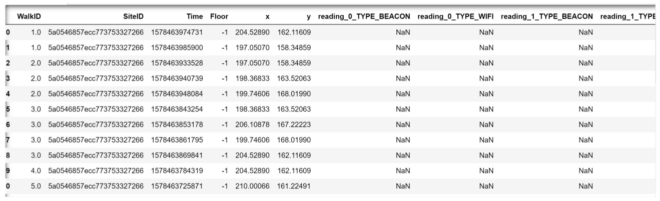
3.3 Data Interpolation

Interpolation of data is the calculation of the aggregate of a point for which two values are available. With the dataset given, the entire process includes interpolation on the label "TIME" and merging of the *waypoint_df* and *train_df_sparce*.

During interpolation of data, the data-set is divided into two parts, prev and after. prev stores the merged *waypoint_df* and *train_df_sparce* with the "Reading Time" <= "Time". after stores the merged *waypoint_df* and *train_df_sparce* with the "Reading Time" >= "Time". The aggregate of their times is stored in separate variables. The interpolated data stores the fraction of the time data. The entire table is stored in *train_df* with the labels: WalkID, SiteID, Time, Floor, x, y, reading_ $<(0-8)>$ _ $<TYPE(BEACON/WIFI)>$. This table would be used for building the model and hence is stored in a .csv file.

3.4 Linear Regression Model

From the file, we remove every row for which every column is NaN as they provide nothing to the calculations. This still gives us plenty of data to build a model around.



	WalkID	SiteID	Time	Floor	x	y	reading_0_TYPE_BEACON	reading_0_TYPE_WIFI	reading_1_TYPE_BEACON	reading_1_TYPE
0	1.0	5a0546857ecc773753327266	1578463974731	-1	204.52890	162.11609	NaN	NaN	NaN	
1	1.0	5a0546857ecc773753327266	1578463985900	-1	197.05070	158.34859	NaN	NaN	NaN	
2	2.0	5a0546857ecc773753327266	1578463933528	-1	197.05070	158.34859	NaN	NaN	NaN	
3	2.0	5a0546857ecc773753327266	1578463940739	-1	198.36833	163.52063	NaN	NaN	NaN	
4	2.0	5a0546857ecc773753327266	1578463948084	-1	199.74606	168.01990	NaN	NaN	NaN	
5	3.0	5a0546857ecc773753327266	1578463843254	-1	198.36833	163.52063	NaN	NaN	NaN	
6	3.0	5a0546857ecc773753327266	1578463853178	-1	206.10878	167.22223	NaN	NaN	NaN	
7	3.0	5a0546857ecc773753327266	1578463861795	-1	199.74606	168.01990	NaN	NaN	NaN	
8	3.0	5a0546857ecc773753327266	1578463869841	-1	204.52890	162.11609	NaN	NaN	NaN	
9	4.0	5a0546857ecc773753327266	1578463784319	-1	204.52890	162.11609	NaN	NaN	NaN	
0	5.0	5a0546857ecc773753327266	1578463725871	-1	210.00066	161.22491	NaN	NaN	NaN	

Figure 2: train_df

3.4.1 Training the Data. The table is first read as a .csv file and displayed to make sure that the data is unaltered. From there, different visualizations like scatter plots, etc., can be tried to see the correlations between the data values. For this model, only 80% of the train set is chosen for training the model. The x-axis is taken as the sensor readings for BEACON and WIFI. The y-axis is taken as the readings for x and y. For building the Linear Regression model, sklearn.metric 's *mean_squared_error*, sklearn.*linear_model* 's LinearRegression, sklearn.preprocessing 's PolynomialFeatures, numpy, panda, and matplotlib in Python are used. Multiple x variables store the sensor readings for different columns and multiple y variables store the 'x' and 'y' readings. A transformer from PolynomialFeatures is used to change the x variables to fit the regression model. The linear regression models are built and stored in different variables for the different combinations of the x and y variables. The coefficients, intercept and coefficient of determination of every variable is calculated. Then the different y values for 'x' and 'y' are predicted and printed. The mean squared error is calculated for every predicted y value compared to the original values. The column that gives the lowest mean squared error for the predicted 'x' and 'y' values is chosen for testing. The model is tested on the remaining 20% data of the train set but this time, only the sensor reading column with the lowest mean squared error is chosen, and is accepted if it returns a low mean squared error.

3.4.2 Testing the Data. The train set is read, cleaned and interpolated similar to the train set. After the *test_df* has been attained, the readings are tested on the linear regression model created. However, this time, we just use the sensor reading that had the lowest mean squared error in the training model. The model is built and the 'x' and 'y' values are predicted which provide us the location of the smartphone. The new mean squared error is calculated. The final readings are stored in a file.

4 RESULTS

The results obtained from this experiment were promising and allowed for a much better understanding of the location prediction problem while simultaneously leaving room for improvement.

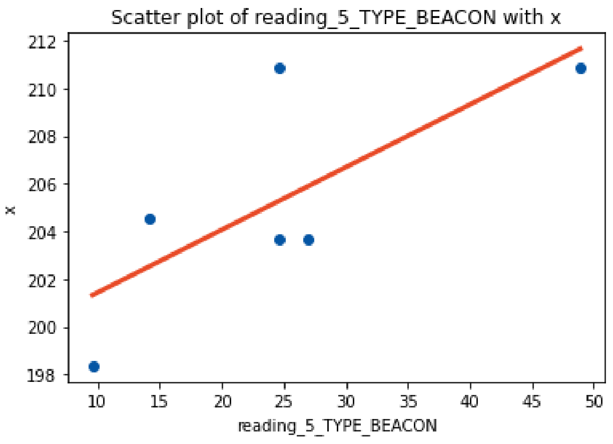


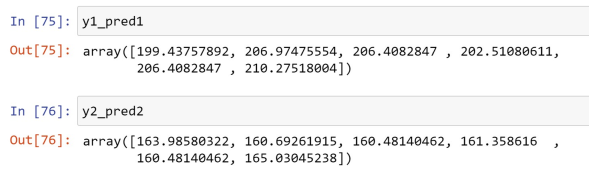
Figure 3: A scatterplot showing the correlation between *reading_5_TYPE_BEACON* and *x*. This plot just shows a minuscule part of the readings whereas the actual plot would contain a lot more marks. This is just for an easier display of the correlation.

```
coefficient of determination: 0.6239259570850062
intercept: 68.86022610671301
coefficients: [-2.62044852 -0.0118727 ]
```

Figure 4: Readings of the linear regression model between a specific sensor data and the location axes.

The linear regression model was first run on 80% of the train data which had been chosen for training. The 'x' and 'y' values were predicted from the linear regression model for each of the sensor readings for BEACON and WIFI, as those were the only two variables allowed to use. The mean squared errors of the different predicted values were calculated. The MSE of x and MSE of y was averaged for each sensor reading and compared. The sensor reading with the lowest average mean squared error was taken as the hypothesis value. In our case, the sensor data with the lowest average mean squared error was *reading_4_TYPE_BEACON*, with an AMSE of 6.974. Then the model was run on the remaining 20% of the train data but this time, the x-axis only contained our hypothesis sensor reading column, *reading_4_TYPE_BEACON*. The predicted 'x' and 'y' values were obtained from the linear regression model and the average mean squared error was calculated. The error came out to be 6.837.

The next step included running the model on the test set. The test set was read, cleaned and interpolated in a similar manner to the train set and the linear regression model with *reading_4_TYPE_BEACON* was run on it. The 'x' and 'y' values were predicted and the mean squared errors and the



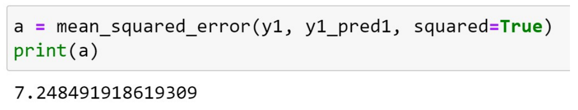
```
In [75]: y1_pred1

Out[75]: array([199.43757892, 206.97475554, 206.4082847 , 202.51080611,
                206.4082847 , 210.27518004])

In [76]: y2_pred2

Out[76]: array([163.98580322, 160.69261915, 160.48140462, 161.358616  ,
                160.48140462, 165.03045238])
```

Figure 5: The predicted 'x' values when built on a model with $reading\_5\_TYPE\_BEACON$**. This just uses a minuscule part of the readings whereas the actual array would contain a lot more data. This is just for an easier display of the correlation.**



```
a = mean_squared_error(y1, y1_pred1, squared=True)
print(a)

7.248491918619309
```

Figure 6: A sample mean squared error that is predicted from the regression model.

average mean squared error was calculated. The AMSE came out as 6.894.

With the average mean squared error being low, the linear regression model with the sensor reading column $reading\_-4\_TYPE\_BEACON$ was accepted. The location of the smartphones could be predicted with an error of 6.894 using this model and hence, the navigation and location in malls could be predicted easily. One of the interesting aspects of this solution is its simplicity. A simple linear regression model was created with a trial-and-error basis for the restricted readings of Bluetooth Beacon and WIFI sensor readings with a low mean squared error.

5 CONCLUSION

This paper proposed an approach to the location and navigation problem in malls using recorded sensor data from Bluetooth Beacon and WIFI. The data-set received seemed daunting at first but after it was read in sections in a systematic way, the data was easier to understand. A simple approach of Linear Regression was taken to tackle the problem. The data-set was cleaned and interpolated so only the waypoint, WIFI and Bluetooth Beacon readings were taken into calculation. A trial-and-error linear regression model was built in which the 'x' and 'y' coordinates of the smartphone was predicted using each sensor data. The mean squared errors and the average mean squared errors were calculated and compared. The one with the lowest AMSE was accepted and tested.

The results were promising with a low AMSE, which gave us a simple model for location prediction using WIFI and Bluetooth Beacon readings. However, this approach could be improved in similar ways. The next task for this research should be to implement a LightGBM model which provides better prediction with lower MSE. Moreover, the predicted and the original pathways should be drawn for better comparison on the Floor Images from the metadata.

REFERENCES

[1] Kaggle (2021) *Indoor Location and Navigation*, Kaggle, https://www.kaggle.com/c/indoor-location-navigation/data
[2] Wikipedia (2021) *LightGBM*, Wikipedia, https://en.wikipedia.org/wiki/LightGBM
[3] Prof. Dr. Antonio Moschitta, Prof. Dr. Jörg Blankenbach (2018) *Applications of Wireless Sensors in Localization and Tracking*, Sensors Networks, $https://www.mdpi.com/journal/sensors/special\_issues/sensors-localization\_tracking$
[4] Amin Ghafourian, Orestis Georgiou, Edmund Barter, Thilo Gross (2020) *Wireless localization with diffusion maps*, ACM, https://www.nature.com/articles/s41598-020-77695-7
[5] Weaam T. EL-Gzzar, Hala B. Nafea, Fayez W. Zaki (2020) *Application of Wireless Sensor Networks Localization in Near Ground Radio Propagation Channel*, IEEE, https://ieeexplore.ieee.org/document/9235118
[6] Marina Md Din, Norziana Jamil, Jacentha Maniam, Mohamad A Mohamed (2018) *Review of indoor localization techniques*, International Journal of Engineering Technology, $https://www.researchgate.net/publication/326982013\_Review\_of\_i-ndoor\_localization\_techniques$
[7] Ali Yassin, Youssef Nasser, Mariette Awad, Ahmed Yassin Al-Dubai (2016) *Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications*, IEEE Communications Surveys Tutorials, $https://www.researchgate.net/publication/311167066\_Recent\_Adv-ances\_in\_Indoor\_Localization\_A\_Survey\_on\_Theoretical\_Ap-proaches\_and\_Applications$

https://drive.google.com/drive/folders/1lXiEecJYaIbCy-LgnZqWlOEG3oivUHHc2