WheelFlow

Benedetta Tessa, Francesca Pezzuti, Franco Terranova, Pietro Tempesti
b.tessa1@studenti.unipi.it, f.pezzuti@studenti.unipi.it, f.terranova2@studenti.unipi.it, p.tempesti@studenti.unipi.it

ABSTRACT

Paths' accessibility is an important problem for user with mobility issues. *WheelFlow* focuses on collecting and sharing geo-localized accessibility information about routes in order to assist wheelchair users. Since data collection is based on crowd-sensing this application is very scalable and accurate, and it is dynamic, as the data gets constantly updated.

1 Introduction

Cobbled city squares, uneven sidewalks surfaces, steep ramps, often pose limitations to wheelchair users, and existing routing applications cannot be directly adapted for people with special needs due to the absence of detailed knowledge of those obstacles; *WheelFlow* is born to offer a solution to this problem.

WheelFlow is a crowd-sourced, Android application where users can search for a path from a source to a destination and receive accessibility information about the path. The core of the application is the collection of vibration data based on crowdsourcing: route contributor users can collect data using their smartphone while they are on the move and share it with a web-server responsible for classifying the data and handling the routing requests. The server receives the raw data and creates an overlay map to show the accessibility of a specific zone or path.

2 Architecture

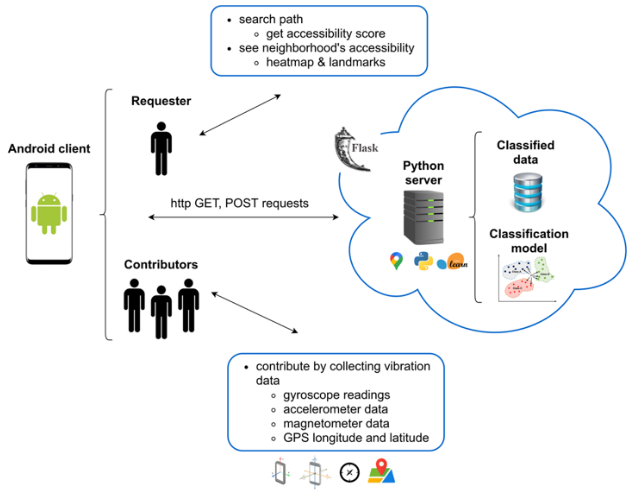

Figure 1: Application architecture

The architecture is composed by a Python server and an Android Client application. The Python server stores the model used to classify the data, handles the interaction with the database and answers the clients' requests allowing users to ask for routing information taking in consideration the accessibility of the locations. The contributors are the core of the crowd-sensed system, and they contribute with surface vibration data collected through the smartphones while using the wheelchair. Any user can behave as contributor while using the application, agreeing on some conditions to respect while using the application (e.g., position of the smartphone).
A KNN classifier is used in order to classify four possible kinds of surfaces that can be encountered in the built environment, depending on accelerometer, gyroscope and the magnetometer data.

3 Data Collection

As aforementioned, the goal of our application is to recognize obstacles through crowdsourcing, therefore we needed to collect was sensor data coming from the GPS, the gyroscope, the accelerometer and the magnetometer.
Such obstacles are uneven roads and steps, but we also decided to register data coming from even roads and ramps, which are facilities that could also be notified to the users.

In order to collect such data, we were provided with a traditional wheelchair that we used in a so-called "assistant-propelled" way, meaning that we needed another person pushing it.
We placed the phone on the upper part of the leg of the sitting person since we thought it was the position with the lowest amount of noise. To make sure the smartphone wasn't moving throughout the duration of the experiment, capturing noisy vibrations, it had been secured using an elastic band.

We built a recording application to collect supervised data, allowing us to start the recording of such sensors, specifying the type of obstacle or facilities we were going to face.
What we obtained was a record containing the following fields: timestamp, latitude and longitude, acceleration along the x-axis, acceleration along the y-axis, acceleration along the z-axis, magnitude along the x-axis, magnitude along the y-axis, magnitude along the z-axis, rate of rotation around the x-axis, rate of rotation around the y-axis, rate of rotation around the z-axis and the correct label (*GAIT*, *UNEVEN_ROAD*, *RAMP*, *STEP*).
Approximately 2 millions of data records have been collected.

3.1 Windowing and Feature Extraction

After data collection, the collected data, ordered by timestamp, had been segmented into windows in order to extract the relevant features of each data window.
This segmentation happened in 2 main phases:

1) Creation of the so-called "macro-windows", each relative to a different measurement. This means that each record in such window belongs to the interval of time in which a specific recording started and ended; therefore they all have the same label.

2) Creation of smaller windows by using a certain size and overlapping, inside those macro-windows, from which we extracted the features. The best values for those parameters were empirically calculated by considering the accuracy of the model.
Such features (calculated for accelerometer, gyroscope and magnetometer) are: magnitude, maximum value, minimum value, difference between maximum and minimum, mean, median, standard deviation, skewness, kurtosis, 25-quantile, 75-quantile, energy, root mean square, number of peaks and several coefficients generated by a Discrete Wavelet Transform.

The total number of features is 118, including the label.

4 Experimental results

We've compared different combinations of pre-processing parameter to find the best classification result, using also different classifiers. The combined parameters are:

- the window size (between 0.5s and 1s with a step of 0.1s)
- the overlapping of the window (between 0s and half of the window size with a step of 0.1s)
- the number of features to select (between 5 and 20)

Talking about classification models, we've compared Decision Tree, Gradient Boosting, AdaBoost, Random Forest, Neural Network, Naïve Bayes, and K-Nearest Neighbors using a Stratified K-Fold cross validation. K-Nearest Neighbors is the model that has reached the best average F1-Score in each test we performed.

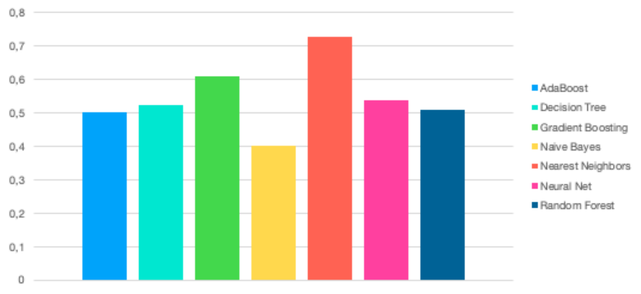

Figure 2: comparison between the different classifiers' F1-Score

In the following graph we can see how the classification results are not much affected by different values of window sizes and overlapping, while the number of features selected has played an important role in our results.

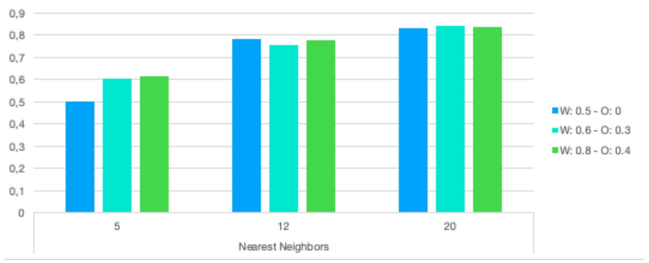

Figure 3: impact of different number of selected features and different window sizes

5 Accessibility Score

In order to provide reliable results, since the machine learning algorithm can provide different outcomes for the same location and since the conditions of the paths may change over time, an adaptive algorithm has been used.

Taking in consideration the results of the classifier in a chronological order, we determined a cumulative score for a macro-zone, which is a set of locations in a radius of 5 meters, used to determine the actual class of this set of locations without relying on a single classification coming from our crowd-sensing system.
Using this approach we achieved the following benefits:

- forgetting outdated measurements which accounts for temporary or permanent changes
- dealing with non-reliable results coming from a single classification
- removing eventual human errors or noise introduced by aids such as different types of wheelchairs

Algorithm 1 Accessibility Score

```
1: threshold = function(location_i)
2: score[location_i] = 0
3: for sample close to location_i do
4:     if entry.class == ACCESSIBLE then
5:         score[location_i] = min(score[location_i]+1, threshold)
6:     else
7:         score[location_i] = max(−threshold, score[location_i]-1)
8:     end if
9: end for
```

$$accessibility = \begin{cases} 1 & \text{if score[location}_i\text{]} >= 0 \\ 0 & \text{otherwise} \end{cases}$$

The accessibility score of a macro-zone has been computed according to the following algorithm. When a user requests for the route from a source to a destination, the Google Maps API provide a route as a combination of multiple waypoints. The average score of macro-zones in a radius of 10 meters from a waypoint along the route will become the waypoint score. A resulting heatmap will be provided to the user by our application showing the different waypoints in different colors depending on their score. Along with these partial routes, a cumulative score for the entire route is provided, calculated as the average score of the waypoints' scores.

5 Conclusions

In order to help increase the accessibility of wheelchair users, different countries have made regulations to be followed during the design, construction and alteration of built environments.

Collecting information of barriers and facilities and their frequent update is though challenging, and therefore we exploited crowd-sensing in our solution.

Using our application, local municipalities can discover in real-time about possible local architectural barriers and handling their reparation in a quick manner.

REFERENCES

[1] Elsevier. 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). S. Mascetti, G. Civitarese, O. ElMalak, C. Bettini. SmartWheels:Detecting urban features for wheelchair users' navigation.

[2] IEEE. 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). J.Edinger, A. Hoffmann, A. Wachner, C. Becker, V. Raychoudhury, C. Krupitzer. WheelShare: Crowd-sensed Surface Classification for Accessible Routing.