P2P Detection of Travel Mode

João Calisto
joao.s.calisto@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract

The detection of the users' travel mode is becoming more relevant due to the ubiquity of mobile devices and the applicability of this technology in multiple contexts. Many solutions can be found in the literature that aims at identifying the transport mode. However, some problems still exist due to the number of variables that negatively impact the system's accuracy, the device's power consumption, detection delays, etc. Therefore, in this thesis, we propose a new solution that combines a common machine learning technique with a P2P Network. This network allows the applications running in each device to exchange information and, consequently, improve the certainty of the classifier. We believe that this solution provides higher confidence levels for each detection while maintaining a near real-time transport identification.
Keywords: Transport mode detection; Classification; Smartphone; Peer-to-peer;

1. Introduction

This work is developed in the scope of the MoTiV project that focuses on investigating the Value of Travel Time (VTT), introducing and validating a conceptual framework for the estimation of the VTT.

MoTiV targets other aspects of VTT, in addition to the economical cost, including individual preferences, motivations and behaviours. A wider definition of VTT can be found when we consider the overall user activities, satisfaction and specific needs, which can be used to better understand travel decision making. Some people might prioritize more economical modes of transportation while others might choose alternatives that offer superior comfort.

The analysis of travellers' decisions is facilitated by developing automated detection mechanisms and eliminating the need for having a user manually inserting information. Automated detection mechanisms must be efficient and accurate to ensure a trustworthy data set while requiring minimal resource usage. Analyzing and understanding the population behaviour is key to improve transport infrastructures resulting in a positive economic and environmental impact, along with social well-being.

The goal in this work is to develop a solution that detects user's travel mode (i.e., travelling on foot, bicycle, train, car, bus) with adequate accuracy (at the minimum 85%, ideally over 90%). This solution consists in an Android application (DetectP2P) that communicates, in real-time and without internet access, with the surrounding instances of the application running in other devices. DetectP2P aims at using the communication between devices to offer more resistance against sensor noise and allowing devices to share mobility knowledge that is then used to improve future decisions.

The sensor readings depend on multiple factors including variables such as the smartphone hardware, how the user transports the device, user-specific behaviour, possible interferences and many more. The readings from one user walking with the device in his pocket are different from the readings of that same user walking with the device in his hand, adding additional acceleration from his arm movement. Noisy data can have a significant impact on the accuracy of the model [1].

Our approach explores one alternative to current solutions that either require a remote server to execute the detection algorithm (without real-time detection) or directly execute it on the device itself (i.e., stand-alone) with limited resources.

In this solution, the machine learning approach used in Woorti [2] is enhanced with mobility data being shared between users in a peer-to-peer environment. Such data includes the broadcast of transport mode predictions in real-time and previous trip validations confirmed by the users. To infer the transport mode, DetectP2P combines the decision from the machine learning classifier, the decision from the surrounding peers and the information obtained from the local and peer validations.

2. Related Work

The detection of travel mode is included in the field of Activity Recognition and provides information that is relevant in multiple contexts such as collecting data to manage traffic and road congestion, improve transport infrastructures, the automation of specific settings in smartphone applications (e.g., user's playlists, advertisements) and many more. Therefore, we can find multiple solutions that try to identify the users' transport mode, particularly, solutions where the whole detection process occurs in the user's device (i.e., stand-alone classification) and solutions that make use of a remote server to receive the collected data and to run the detection algorithm (i.e., remote classification).

2.1. Stand-alone Classification

The improvements associated with mobile devices technology in the last years, specifically hardware components (e.g. CPU, RAM, storage), led to the ability to run more complex and resource-intensive applications in the average smartphone [3]. The main advantage of this approach is that there is no data exchange between the application running in the user's device and a remote server, allowing the classification to be performed in real-time.

Martin et al.[4] proposed a solution that achieves an accuracy of 96.8% using the GPS (sampled every second) and the accelerometer (sampled five times per second). They explored and evaluated three different methodologies (movelets, k-nearest neighbours and random forests) to distinguish between five travel modes: walking, biking, bus, car and rail. In order to adapt the classification process to the smartphone limited resources, they also investigated two feature reduction techniques: principal component analysis (PCA) and recursive feature elimination (RFE). It was claimed that these methods could be utilized in a smartphone without substantial burden on battery life. However, no power consumption data was presented to support this claim. Additionally, their evaluation consisted exclusively of trips with a single mode of transportation. Thus, they do not apply their solution in trips with a transition between different modes.

Sauerländer-Biebl et al.[5] investigated and evaluated one stand-alone solution based on fuzzy rules, with GPS and accelerometer data, presented in [6]. They use a set of rules, that is manually defined through the analysis of test data, to establish a membership degree (from zero to one) to each class. These rules are based on parameters related to speed, acceleration, turning angles, etc. They obtained a high accuracy (98%) for car trips classification but had rather low results (75%) for the overall classifications. Methods based on fuzzy logic are relevant in the classification of incomplete and uncertain information (e.g., facial pattern recognition, weather forecasting, etc.) which is also the case of transport mode detection [7]. However, to correctly identify modes of transportation with such varying conditions, it is extremely helpful to use a machine learning approach that is able to adapt and learn as those conditions vary. Furthermore, the authors don't provide data related to the device's power consumption.

Chen at al.[8] present Mago, a system that uses the accelerometer and the Hall-effect magnetic sensor to distinguish between 7 classes (stationary, bus, bike, car, train, light rail and scooter). The main contribution of this article is the use of the magnetic sensor to infer the user's transport mode. Mago requires a smartphone with a Hall-effect sensor with a sensitivity lower than 0.3 µT/LSB and a minimum sampling rate of 100Hz, which is not currently available in a big portion of devices (e.g., iPhone 6, iPhone 5s, Google Nexus 5, etc.). Additionally, this solution is not able to identify the most common physical activities like walking or running.

2.2. Remote Classification

Multiple solutions choose to implement the classification algorithm in a remote server (e.g., in the cloud). In this approach, the application running in the user's smartphone (or dedicated device) is only responsible for data collection. This data is transferred to a remote server where the respective classifier is executed.

Stenneth et al.[9] use additional data that represents bus locations, rail lines and bus stops spatial data (available to the public in real-time) to distinguish between motorized transports (e.g., car, train and bus) while still identifying other modes such as bike, walk and stationary with an overall accuracy of 93.5%. This solution is limited to the geographical areas that offer transport information in real-time. Additionally, the authors don't provide power consumption information but the exclusive use of GPS data might drain the battery lifetime of the device.

Das & Winter [10] propose a hybrid knowledge-driven framework based on fuzzy logic and neural networks with the objective of adapting to varying conditions, which is a problem of solutions based in exclusively fuzzy models. This approach achieves an accuracy of 83% while being able to explain the reasoning process for a given choice and to tolerate noisy data. However, this solution does not identify relevant travel modes (e.g. car, bike, running) and its exclusive dependence of the GPS sensor might significantly drain the device's battery. Additionally, it uses features related to the proximity to bus, train and tram network (routes) which limits the system to certain regions.

Hemminki et al.[11] used a purely accelerometer-based approach to distinguish between 7 different modes (stationary, walk, bus, train, metro, tram, car) with an accuracy of 84.9%. They developed an algorithm to estimate the gravity component of the accelerometer readings and consequently derive the component that represents the user motion. This algorithm has the disadvantage of being sensitive to noisy data, which in the case of smartphone-based data collection is quite frequent due to the constant variations of the device orientation and interferences related with the handling of the device. Additionally, the authors report a high latency in the classification when the user changes to a motorized transport, which could be solved with the use of other sensors (e.g., GPS).

3. Background

DetectP2P imports the Trip Detection module of Woorti, a mobile application used to understand the value of travel time perceived by the user. In this section, we briefly describe how this module works in order to better understand how it can be improved in the following sections.

The GPS is used to obtain the coordinates that represent the user's location. From multiple GPS coordinates, the user's travel speed is inferred by dividing the distance travelled by the time interval. The accelerometer is used to obtain the user's acceleration along three axis.

These raw values, that represent acceleration and speed, are grouped into trip segments with 90 seconds. The raw data from each segment is processed in the Preprocessing module to calculate the features of the transport mode. These features represent the characteristics of the trip, such as average, maximum and minimum speed, percentage of acceleration values between 0.3 m/s² and 0.6 m/s², percentage of acceleration values between 0.6 m/s² and 1.0 m/s², etc. The complete set of 23 parameters that describe each segment can be consulted in the article where Woorti's methods were developed [2].

After processing each segment, the resulting parameters are fed to a random forest classifier that evaluates the segment and stores the resulting list of probabilities correspondent to each possible transport mode. When the user is stationary (i.e., distance and accelerations smaller than a defined threshold) the trip ends, and the post-processing phase begins.

In the post-processing phase, strong segments (i.e., segments with a high walking probability) are used to split the sequence of segments into smaller sequences. These walking segments are easily identifiable due to the high acceleration values along the three axis. This allows us to identify where each mode of transport sequence starts and finishes. Finally, each sequence is evaluated again and the final mode of transport for each sequence is obtained.

The random forest classifier was previously generated in the training phase, with 537 trips (corresponding to 265 hours) obtained from a group of volunteers collecting data in their daily routine. Those trips were separated into a training and a testing data set to train the classifier. Then, the classifier model was generated and imported to the smartphone application.

4. Solution & Implementation

DetectP2P collects data from the smartphone sensors and evaluates it with the machine learning approach developed in Woorti, where a mathematical model based on training data (i.e., GPS and acceleration metrics) is used to make an initial prediction on the current mode of transport. The validation history of the local user is used to adjust this initial prediction. DetectP2P communicates with the closest devices via Bluetooth to exchange stored user validations and transport mode predictions in real-time. Then, the knowledge obtained from the local and external validations is used to make a prediction based on the route taken. An additional prediction is created by analysing the predictions from the surrounding devices. By combining the three predictions (i.e., adjusted classifier prediction, path score, peer prediction), a final decision is taken.

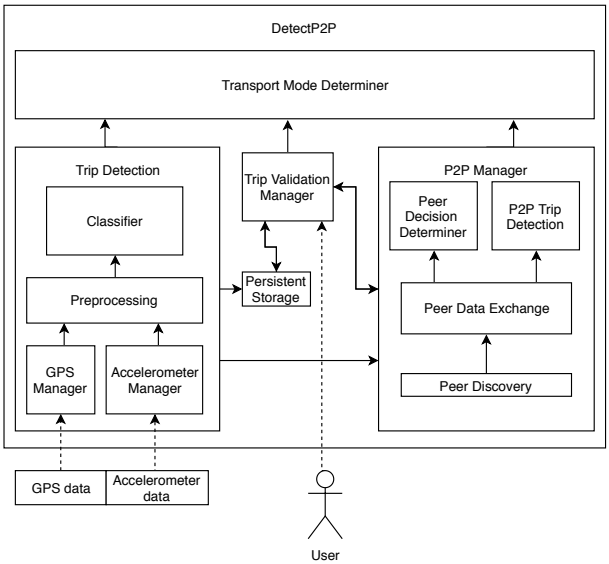

Figure 1: DetectP2P architecture and information flow.

The final decision towards the user's transport mode is taken by the Transport Mode Determiner. This module gathers the output of three other main modules: Trip Detection, Trip Validation Manager

and P2P Manager.

Trip Detection (developed in Woorti): Detects the start/end of each trip and makes a prediction based on the smartphone sensors data (i.e., GPS and accelerometer).

Trip Validation Manager Collects user validations, storing information that relates the trip with the classifier's prediction and the real mode validated by the user.

P2P Manager: Responsible to establish connections and share information with the other DetectP2P instances within the range of the user. This module takes a decision based on the surrounding smartphone's decisions.

4.1. Trip Validation Manager

When the Trip Detection Module detects and evaluates a trip, it saves the trip structure (i.e., sequences of segments with the corresponding evaluation) in the device's persistent storage (i.e., local database). Later, the user can verify the trip and confirm the correct mode of transport. This validation, representing the real mode of transport, is stored together with the classifier's prediction and the respective trip coordinates in the file system. Coordinates are needed to infer the path of the respective trip, which enables the system to later recognize similar trips with equal conditions.

These trip validations allow us to analyze the performance of the classifier, but the validations from a single user might not be enough to extract accurate metrics. Thus, the validations of the other users are also stored in the local database. Those validations are obtained by the P2P Manager while communicating with other devices.

In order to avoid repeated validations, each validation is associated with a unique user and trip identifier, generated by the original device where the trip was detected. To generate a unique representation of the trip, we create a tripId that appends the following data: start timestamp (ms), end timestamp (ms), an average of the absolute acceleration values and distance travelled (m).

Additionally, this module creates a confusion matrix (i.e., relation between observed and predicted mode) that summarizes the results of the validations. From the confusion matrix, we can obtain the true positives, false positives, false negatives and true negatives, relative to the classifier's performance. Then, we can infer more metrics to evaluate the classifier, such as the false positive rate, false negative rate and precision. These metrics represent the likelihood of one mode being confused with another one. That relation allows the system to adjust the classifier prediction (see Subsection 4.3).

4.2. P2P Manager

In this section we describe the peer-to-peer component of our solution, particularly the discovery of peers, communication between devices, an alternative method to detect trips and, finally, the classification of a trip based on peer data.

4.2.1 Peer Discovery Manager

Before starting to share information between devices, the connections must be established. This is the responsibility of the Peer Discovery Module. It discovers and connects to other devices via Bluetooth [12], a limited range communication link that allows the devices to directly connect with each other.

At any moment, the user is between three possible states: travelling, stationary (i.e., not travelling), waiting (i.e., travelling but currently stationary). When the user is stationary, DetectP2P keeps trying to discover new peers in order to detect the start of a new trip. In the waiting state, DetectP2P needs to be able to decide if the next state is still (i.e., the trip ends) or travelling (i.e., the trip continues). The predictions of other peers are used to decide which state follows the waiting state. In the travelling state, we use the predictions to identify the current mode of transport being used. Thus, we need the discovery process to connect with users sharing the same transport mode and obtain their predictions. Therefore, as the list of peers can be constantly changing (e.g., user's moving in and out of each other's range), the discovery process must be running during the whole life cycle of the application.

To keep track of the surrounding users, this module keeps a list of connected peers and the respective connection timestamp. This list is updated whenever one device enters or exists the user's range. When the connection with one peer is established, the devices will be able to share information until one of them disconnects.

4.2.2 Data Exchange

There are two classes of information being shared: real-time predictions and trip validations. Real-time predictions are the local classifier decisions that are broadcasted to every peer when one segment is evaluated. These predictions consist of a list of probabilities associated with each possible transport mode. The trip validations are managed by the Trip Validation Manager and are sent to other peers in order to share the knowledge relative to the classifier's performance. These validations include the classifier prediction, the corrected mode and the trip GPS coordinates. The validation exchange occurs when two users initiate a connec-

tion.

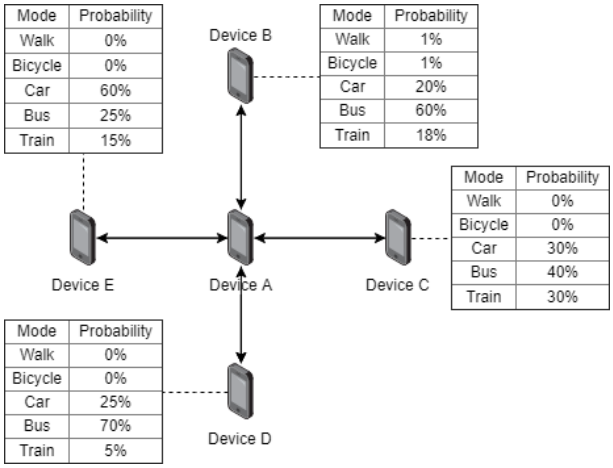

Figure 2: Perspective of device A in a network of 5 devices.

Regarding real-time predictions, every time the random forest classifier evaluates a segment (i.e., a trip fragment with 90 seconds), the result of the evaluation is sent to the P2P Manager that broadcasts the data to every connected peer. In Figure 2, we can visualize how the predictions are broadcasted. Each device broadcasts its prediction and receives the predictions of the connected devices.

Regarding trip validations, each peer shares all known validations, including the validations performed by the local user and the validations from every device that has ever established a connection. When two devices establish a connection, the validation exchange begins. Each device shares the list of *tripIds* corresponding to every validation stored in its database. Given that each validation is only associated with one unique *tripId*, the other device can filter the validations already acquired and request only the *tripIds* that it does not contain. Then, following that request, all the corresponding validations are sent. The objective of this protocol is to minimize the data being transmitted by avoiding the transfer of repeated validations.

4.2.3 P2P Trip Detection

While the Trip Detection module is the main responsible for detecting the start and end of each trip, the P2P Trip Detection is also able to perform this task when needed. This includes the cases when the Trip Detection module fails to detect the trip due to lack of GPS signal (e.g., disabled to save battery). This is relevant to collect travelling data when the user wants the device to run on low power consumption (i.e., without activating the GPS).

This module detects the start and end of a trip purely based on the information received from the surrounding peers. The real-time predictions received from other peers, indicating that they are travelling, can be used to detect the beginning of our own trip.

There are four possible scenarios when receiving external predictions:

1. Device A is stationary and connects with a stationary device B.
2. Device A is stationary and briefly connects with a moving device B.
3. Device A is moving and briefly connects with a stationary device B.
4. Device A is moving and connects with device B that is also moving.

By filtering the first three scenarios, we can infer the last scenario, where the device is moving, and recognize the start of a trip. This filter consists of analyzing the predictions from devices with a connection time superior to a threshold of 5 minutes. The objective of the 5-minute threshold is to distinguish between the devices that only intersect for a small period of time and those that are actually travelling with us.

In scenario 1, there are no moving devices sharing their predictions so there is no indicator of a trip starting. In scenario 2, device B might connect with device A, but the connection time will be inferior to the threshold because the other user will eventually move away and break the connection. In scenario 3, the user is moving but, because there are no moving devices sharing predictions, DetectP2P does not have any information to confirm the start of a trip. In scenario 4, the current device has predictions from a moving device. If the connection time is superior to the threshold, it indicates that the current device is keeping up with a moving device. Thus, DetectP2P can assume that a trip has started.

To detect the end of a trip, we need to infer scenario 1. This is done by finding devices connected for a time superior to the threshold, without broadcasted predictions (i.e., stationary devices). By knowing that we are close to a stationary user for a given period of time, we can assume that we are stationary too. When there are no peers connected, we also end the trip.

4.2.4 Peer Decision Determiner

When the application detects the end of a trip, the decision process in the Peer Decision Determiner begins. The ending of a trip is detected by the main Trip Detection module or by the alternative P2P Trip Detection module.

The first step is to filter valid peers. To do this, we need the connection timestamp associated with each peer. The connection time is used to filter the devices that are sharing the same transport. If the connection time is inferior to the 5-minute threshold, we will discard the information from that peer as it is not enough to assure the respective user is travelling by the same transport mode.

Each valid peer can have one or more evaluations broadcasted, so the next step is to associate the predictions with each trip part, based on the start and end timestamps. From each prediction sequence, results one single prediction representing the arithmetic mean of the predictions considered for that specific trip part. At this point, we have, for each peer, a list of intermediate predictions. Each intermediate prediction corresponds to the travel mode probabilities associated with a specific trip part.

The next step is to calculate the weighted arithmetic average for all the peers to obtain the decision from the P2P Manager for each trip part. The weight of each peer in the decision corresponds to a confidence factor (CF), calculated with Equation 1. The confidence factor is based on the peer's device accuracy, which is calculated by dividing the number of correctly predicted trips (CP) by the total count of detected trips (T). If one of the peers does not have a minimum amount of 10 total trips (i.e., average number of trips collected in a daily routine), that peer is not considered.

$$CF = 1 + (\frac{CP}{T} * 10) \quad (1)$$

Finally, this module returns its output to the Transport Mode Determiner, consisting in one prediction for each trip part from a given trip.

4.3. Transport Mode Determiner

When the application detects the end of a trip, the Transport Mode Determiner is called to take the final decision on the detected mode of transport. This module gathers the output of the other three modules: Trip Detection, Trip Validation Manager and P2P Manager.

From the list of probabilities generated by the random forest classifier (i.e., in the Trip Detection module), we consider the first, second and third modes with the highest probability. The first, second and third mode with the highest probability are referred to as M1, M2 and M3, respectively.

In the first step, we take into consideration the false positive and the false negative rates obtained from the local validations. In Algorithm 1 we describe the adjustment of the classifier prediction according to the validation history of the user. The input needed to perform the adjustment corresponds to the local confusion matrix (T) calculated by the Trip Validation Manager, the probabilities map (D) for each mode generated by the Trip Detection module and the count of occurrences for each mode. From the confusion matrix, we obtain the percentage of occurrences (P) where M2 and M3 were incorrectly identified as M1. Then, we decrease the probability of the three modes in equal portions (i.e., a third of P) and distribute P according to each mode frequency.

Algorithm 1: Adjustment of the classifier decision

Input: T, confusion matrix of the local validations, T[classified][observed]
D, map with probabilities by mode
C_1 , count of occurrences for M_1
C_2, count of occurrences for M_2
C_3, count of occurrences for M_3

1 $P = T[M_1][M_2] + T[M_1][M_3]$
2 $F_i = C_i/(C_1 + C_2 + C_3)$
3 $D[M_i] = D[M_i] - P/3 + F_i * P$

The next step is to iterate through all the trip validations (i.e., local and external validations). We compare the trip coordinates with the coordinates of each validated trip. We consider that there is a match between two trips when at least 80% of the coordinates from the trip being evaluated are found within the trajectory of the validated trip. A 20% error margin is given to account for accuracy errors and differences in the trip start/end point (e.g., one trip might have more coordinates because it was detected earlier). To compare trips, for every coordinate (C) in the trip being evaluated, we check if it is between any two points (A and B) that belong to the validated trip, while taking into consideration a margin for the GPS accuracy (ACC) of 13 meters, which is the maximum deviation found in the analysis of GPS locations with Android [13]). To verify if point C is between A and B, the following equation is tested.

$$distance(A,C)+distance(C,B) < distance(A,B)+ACC \quad (2)$$

When there is a match between the validation and the current trip, we update the count of occurrences associated with each considered mode of transport. Then, we generate a prediction by assigning a probability to each mode that corresponds to their frequency along that route.

Finally, to merge the adjusted initial prediction, the path score and the prediction from the P2P Manager (i.e., predictions from the surrounding peers), DetectP2P calculates the arithmetic average of the three scores. If the count of peers is

superior to 4, we choose the public transport with the highest probability. In the case where we have transport modes with a small probability difference (i.e., smaller than 5%), we choose the mode that was most used by the user according to the local validations.

5. Results & discussion

DetectP2P can be used in multiple scenarios and the results obtained vary with numerous factors such as travelling alone, travelling with other users, the number of validated trips, the route taken, etc. Thus, to analyse how the application performs on these scenarios we perform the following evaluations:

- Accuracy of the local decision (i.e., random forest classifier) and its variation with users' validations.
- Tendency of the users to repeat trips.
- Processing time to calculate the path score.
- Comparison between local and peer predictions.
- Accuracy of DetectP2P with a group of users travelling together.

We requested 5 people to use DetectP2P for two weeks to collect enough data from their routines. This evaluation was made with five Android devices: Xiaomi Mi A2, Xiaomi Mi A1, Samsung J3, Motorola XT1068 and Huawei Y6. These devices have different hardware from different manufacturers.

Our evaluation was divided into two phases. In phase 1, the users were asked to use DetectP2P in their daily routines, while validating trips at the end of the day. Then, in phase 2, the users were asked to travel together allowing us to collect the data that represents how the system behaves and evolves in a real scenario.

5.1. Adjustment Process

We start by evaluating how the application performs when the user is travelling alone. In this situation, the final decision is based in the local classifier (i.e., Woorti's decision) and it is adjusted with the local validations (with Algorithm 1).

To evaluate this scenario, we considered the trips collected by one of the users (userA) in the first week, corresponding to a total of 105 trips, particularly: walk (64), car (24), bus (5), train (7), bicycle (5). With the collected trips, we proceeded to simulate the adjustment process for different sets of validations as seen in Figure 3 to compare the results obtained with multiple sets. For each set, we ran the adjustment process and collected the results. We considered the validations from userA by their chronological order, with the final set corresponding to the 105 trips.

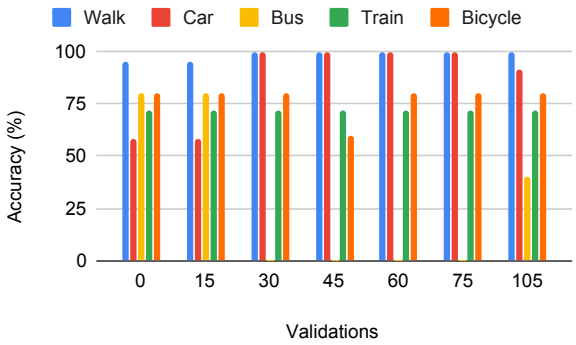

Figure 3: Evolution of the accuracy with the local validations.

Figure 3 shows that in the first set of validations, the accuracy values remain unchanged because there are not enough validations to perform the adjustment. In the second set, we see an increase in the walk and car modes while the bus significantly decreases. This is explained by the preferences of the users towards the walk and car modes. In the three occurrences where the classifier chose the bicycle instead of the walking mode, the system looked into the bicycle false positives (4.8%) and distributed that probability according to the walking and bicycle frequencies, increasing the probability for walk, which is enough to change the predicted mode. The same event occurs in the car trips that were incorrectly identified as bus, with the bus classifications being adjusted to car due to the observed user preferences.

When the total of 105 trips is considered, the accuracy for car drops to 91.6% due to the bus and train trips that are added to the validations. In this situation, the frequency of the car drops to 66.7% which is enough to maintain 2 incorrect bus predictions in car trips. Consequently, we also observe an increase in bus accuracy to 40%.

5.2. Path Score

In this section, we consider the total amount of 316 trips collected by 5 users. The objective of this evaluation is to understand how the increasing amount of validations provides relevant data to infer the mode of transport and analyse the impact, in terms of processing power, that the calculation of the path score has on the trip detection.

5.2.1 Analysis of the Path Relation

In this evaluation, we compare the routes of all collected validations to observe the number of trips that would be improved by integrating the path score in DetectP2P's final decision. Every repeated trip (i.e., a trip with an equivalent route

found in the validations) has an increased probability towards the mode validated in the original trip. Within the repeated trips, we also analyse trips repeated from peers where the original trip was validated by other users.

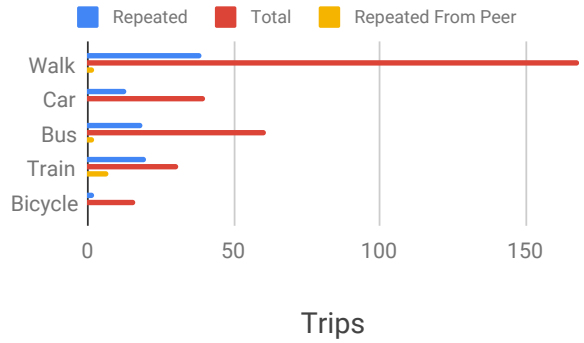

Figure 4: Visualization of the amount of trips with equivalent routes.

In figure Figure 4 we show the portion of trips with equivalent routes in comparison with the total amount of collected trips. By iterating through the total of 316 trips, we searched for trips with equivalent paths. For each trip, we only compare it with the previously iterated trips. For example, to verify if the third trip is repeated, we compare it with the first and second validated trip. We obtained a count of 93 (29.4%) repeated trips, where 39 correspond to walk, 13 to car, 19 to bus, 20 to train and 2 to bicycle. Among the repeated trips, there are 11 (2.8%) occurrences where an equivalent trip was obtained from a peer (i.e., external validations), while the remaining events correspond to equivalent trips found within the local validations.

5.2.2 Processing cost

Given the time requirements for the output of a transport mode decision, we need to analyse the computation cost to obtain the path score. This evaluation was performed on a Xiaomi Mi A2, featuring a Qualcomm Snapdragon 660 and 3GB of RAM. To perform this test, we considered multiple sets of validations (50, 100, 300, 600, 1000, and 2000) and a trip with 360 locations. Each set contains small trips (50%) with 120 locations, medium trips (40%) with 300 locations and large trips (10%) with 720 locations.

Figure 5 shows how the execution time needed to attribute a path score increases with a higher amount of validations. We do not see a linear growth of the execution time because each validated trip has different coordinates. Each validation is discarded when the trip comparison reaches the threshold of distinct coordinates (20%). Thus, the time needed to discard each validation is different across the set. Even considering the highest

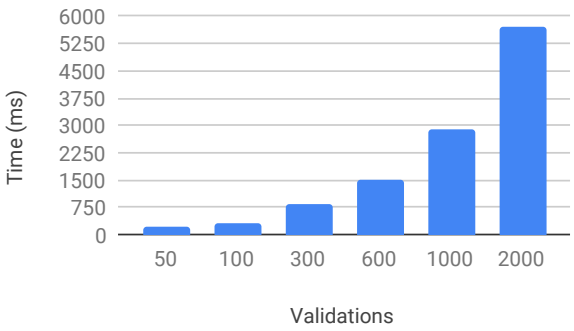

Figure 5: Execution time (ms) to calculate the path score.

value tested (i.e., 2000 validations), we obtained a processing time inferior to 6 seconds which is acceptable considering our requirements for the output of a decision in 30 seconds.

5.3. P2P Evaluation

In this evaluation we test the transport mode detection in a P2P environment where the users travel within a group and their devices communicate with each other during the trip. The objective is to see how the classifier predictions vary between different devices and evaluate our solution, with the contribution from every module, in a scenario where a group of users are travelling together.

5.3.1 P2P Manager Decision

Figure 6 compares the local prediction (relative to the real mode used) of one device with the prediction obtained from the P2P Manager, where the weight of each peer is based on the trips collected in phase 1 (i.e., users travelling alone). Each value in the x-axis represents a single trip without any relation with the others.

In car trips, we observe that the peer prediction is higher than the local prediction in 19 (48.7%) occasions. If we only consider this prediction to take the decision, the peer predictions would correct the decision to car in 4 (10.3%) occasions (trip 6, 13, 33, 34) and change the decision to an incorrect mode in 1 (2.6%) occasion (trip 29). In bus trips, the predictions from the peers are higher than the local one in 18 (56.3%) occasions. The decision would change to the correct mode in 1 (3.1%) occasion (trip 45) while the correct decision is not changed in any scenario. In train trips, the peer predictions increase the probability in 5 (50%) occasions. The decision would be corrected in 1 (10%) occasion (trip 65). While walking, the predictions from the other devices are higher in 7 (41.2%) occasions and adjust the decision to the correct mode in 1 (5.9%) case (trip 68). In bicycle trips, the peer predictions are higher in 5 (62.3%) occasions and there was not any recorded event where the deci-

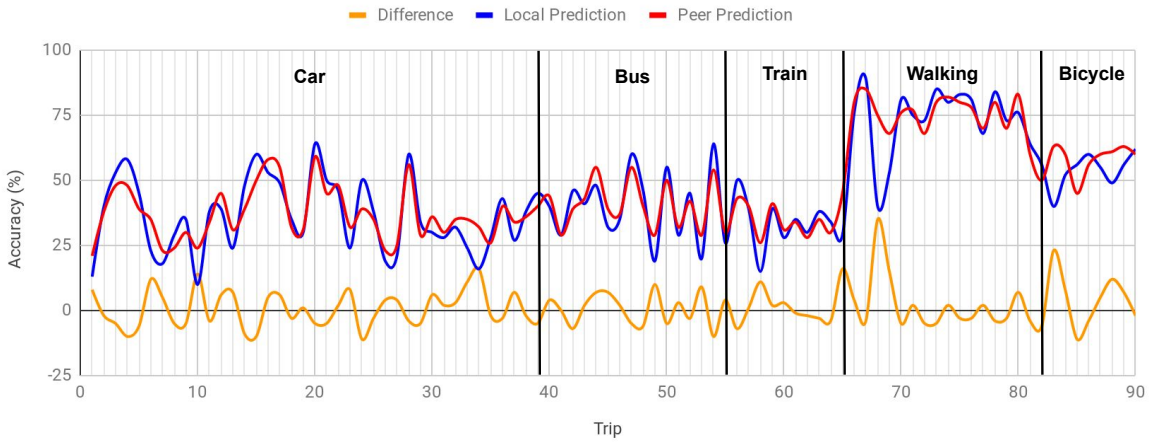

Figure 6: Comparison between local prediction and peer prediction.

sion would change.

While the difference between the local and peer decision is not frequently higher than 15%, we can observe that when the local probability significantly drops, the peer decision offers a slightly better probability. This is relevant because when, due to the smartphone conditions, the local prediction is affected, the peer predictions can compensate with higher probabilities towards the correct mode.

The cases where the peer prediction is lower than the local one are not necessarily disadvantageous. For example, in a case where the local probability corresponds to 60% and the peer probability corresponds to 50%, we observe a 10% drop but the decision is still correct with a smaller confidence. We can slightly decrease the confidence of the correct predictions, but while offering an overall better percentage of correctly detected transports by compensating the low probabilities of the local decision.

5.3.2 Final mode decision

In this evaluation, the contribution from every module is included in the final decision. We use the results obtained in phase 2 of the evaluation (i.e., users travelling together) where each device already contains the validation history from its user obtained in phase 1.

	Predicted Mode					Events
Real Mode	Walking	Car	Bus	Train	Bicycle	
Walking	100.0	0.0	0.0	0.0	0.0	64
Car	0.0	80.1	16.0	3.9	0.0	156
Bus	0.0	14.0	84.4	1.6	0.0	64
Train	0.0	2.5	0.0	97.5	0.0	40
Bicycle	13.9	0.0	0.0	0.0	86.1	36

Table 1: Final confusion matrix (in %) obtained from the results of every user

Table 1 shows the confusion matrix that sums the final decisions obtained by every user. In this evaluation we considered 4 users travelling together for a total of 90 unique trips, specifically walking (16), car (39), bus (16), train (10) and bike (9). The results obtained from each user were aggregated to generate this matrix which means that each trip is considered four times (i.e., one evaluation from the perspective of each user). Overall, we correctly classified 87.7% of the trips. It is important to note that a major part of the experiments was made in an urban environment.

Walking trips were detected with 100% accuracy. By analysing the modes used by each user, we observed that the walking mode is used with higher frequency when compared to the other modes, particularly with the bicycle which is the mode frequently confused with walking. Thus, when in doubt between bicycle or walking, the adjustment process increases the walking probability. Additionally, in the cases where the device can not accurately detect that the user is walking (i.e., user briefly stops, picks up the device from his pocket, etc.), the peers will still have strong walking predictions.

While the adjustment process favours the walking mode, we still observe a satisfactory accuracy (86.1%) towards the bicycle mode. This is explained by the predictions from the surrounding peers, which are strong enough (i.e., usually higher than 55%) to guide the decision in the direction of the bicycle.

Regarding car trips, we obtained a reasonable accuracy of 80.1%. The decision between car and bus highly depends on the user's preferences as the characteristics of these modes are quite similar, particularly in an urban environment where the travel speed is relatively low.

Similarly to the car, the results obtained for the bus vary according to the preferences of the user.

In our experiments, we obtained an accuracy of 84.4%. However, only a small portion (6) of the bus trips consisted of repeated trips going through the same path of another trip found in the user validations.

We can observe the relevance of the path score in train trips, with an obtained accuracy of 97.5%. The path represented by the train rails goes through multiple train stations. If one user validates a long train trip, it will help in the identification of all the train trips that go through smaller portions of that route.

6. Conclusions

This work introduces a new technique to detect trips and identify the chosen mode of transport. We used Woorti's Trip Detection module, where a random forest classifier is used to classify features extracted from accelerometer and GPS data. While the former solution runs in a stand-alone approach, we implemented a protocol to exchange information with the closest devices via Bluetooth. In our solution, each device can share its local decision with the surrounding instances of the application. Additionally, with a mechanism to validate trips, the devices can share validated trips with each other, obtaining relevant knowledge that can be used in their future decisions.

Overall, we correctly classified 87.7% of the trips. It is important to note that a major part of the experiments were made in an urban environment. This conditions significantly impact the results obtained, particularly for car trips where the speed limit is quite low (i.e., maximum of 50 or 80 km/h) and there are multiple traffic lights in the trip, which results in stop times and speed metrics very similar to the bus trips. Given this conditions, the overall accuracy obtained is quite interesting.

This solution adapts to the users' preferences and and takes into consideration the most frequent routes taken by the users. We believe that, with more validations from the users, the tendency would be to achieve higher accuracy values (i.e., more than 90%).

References

[1] Robert Ross and John Kelleher. A comparative study of the effect of sensor noise on activity recognition models. In *Evolving Ambient Intelligence*, pages 151–162, Cham, 2013. Springer International Publishing. ISBN 978-3-319-04406-4.

[2] Constantin Zavgorodnii. DTBM - detecting travel and behavioral mode. Master's thesis, Instituto Superior Técnico, October 2018.

[3] Qiwei Han and Daegon Cho. Characterizing the technological evolution of smartphones: Insights from performance benchmarks. In *Proceedings of the 18th Annual International Conference on Electronic Commerce*, pages 32:1–32:8. ACM, 2016. ISBN 978-1-4503-4222-3.

[4] Bryan D. Martin, Vittorio Addona, Julian Wolfson, Gediminas Adomavicius, and Yingling Fan. Methods for real-time prediction of the mode of travel using smartphone-based gps and accelerometer data. ISSN 1424-3210.

[5] Anke Sauerländer-Biebl, Elmar Brockfeld, David Suske, and Eric Melde. Evaluation of a transport mode detection using fuzzy rules. *Transportation Research Procedia*, 25:591 – 602, 2017. ISSN 2352-1465.

[6] István Pál and Anke Sauerländer-Biebl. Automatische multimodale verkehrsmoduserkennung und situationserfassung mit hilfe von fuzzy-regeln. In *3. Interdisziplinärer Workshop Kognitive Systeme: Mensch, Teams, Systeme und Automaten*, 2014.

[7] Harpreet Singh, Madan Gupta, Thomas Meitzler, Zeng-Guang Hou, Kumkum Garg, Ashu Solo, and Lotfi A. Zadeh. Real-life applications of fuzzy logic. *Advances in Fuzzy Systems*, 2013, 08 2013.

[8] Ke-Yu Chen, Rahul C. Shah, Jonathan Huang, and Lama Nachman. Mago: Mode of transport inference using the hall-effect magnetic sensor and accelerometer. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*

[9] Leon Stenneth, Ouri Wolfson, Philip S. Yu, and Bo Xu. Transportation mode detection using mobile phones and gis information. ACM, 2011. ISBN 978-1-4503-1031-4.

[10] Rahul Deb Das and Stephan Winter. Detecting urban transport modes using a hybrid knowledge driven framework from gps trajectory. *International Journal of Geo-Information*, 5:207, 11 2016. doi: $10.3390/\mathrm{ijgi5110207}$.

[11] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13. ACM, 2013. ISBN 978-1-4503-2027-6.

[12] Bluetooth SIG. Bluetooth, 2019. http://www.bluetooth.com.

[13] Krista Merry and Pete Bettinger. Smartphone gps accuracy study in an urban environment. *PLOS ONE*, 2019.