Phone Position Recognition Challenge

Kavita Chopra

Institute of Computer Science
Rheinische Friedrich-Wilhelms-Universität Bonn, Germany
s6kachop@uni-bonn.de

Abstract

Research has shown that the capabilities of the sensors provided by today's smartphones have the potential to infer the exact phone position with reference to the user's body. The knowledge of the exact position (e.g. hand, jacket pocket, shoulder bag) could be beneficial in enhancing the quality of context-aware mobile applications. Furthermore, any resource-intense application or service running in the background could operate at a higher efficiency if the phone position was known since it could be automatically deactivated if the mobile device was being carried in unfavorable positions. In an aim to develop a smartphone discovery service, we found in the course of our thorough analysis that the challenges consist of choosing the right combination of sensors, an optimal setting of their modalities, and finally an optimal set of positions which are distinct enough so that they can be distinguished by a moderately complex classifier. We used those considerations to develop our prototype for the Android-powered Nexus 5 supporting 5 positions, of which 3 are sensed in a non-idle physical context of the user.

1 Introduction

Smartphones owe their widespread popularity to the surging number of mobile applications which become ever more sophisticated with the employment of the different types of sensors that are available in today's mobile devices. The capabilities of the sensors and their steady improvement have given rise to the emergence of context-aware mobile applications which aim to provide a more personal and enhanced user-experience by automatically adapting the application's behavior to the discovered context. However, studies have shown that context-aware applications do not always live up to their declared promises and mostly miss their objective due to the fact that the context is sensed in unfavorable positions of the mobile device (Diaconita et al., 2014). Therefore, recent studies have proposed that a position discovery service could be used as a complement to context-aware applications in order to optimize their quality and reliability (Alanezi, Mishra, 2013). Furthermore, the results from a position discovery service could help to save battery power by automatically switching off background services which might include sensor activities or other calculation processes whenever the smartphone was in an unfavorable position. Our aim was to develop a prototype of a smartphone position discovery service for which we considered the Android-powered Nexus 5 device, a recent model with a widely used operating system.

2 Methodology

The methodology of our approach formed an essential part prior to the implementation of the position discovery service for the Nexus smartphone. It involved the analysis of the resources at our disposal and included major design decisions for parameters that we could gain control over. The methodological steps towards our final position discovery service comprised of the raw data collection, visualization and analysis for selecting the training data and positions that would be supported by our final application, the generation of the classifier and last but not least the integration of the classifier into our prototype of the position discovery service.

2.1 Data Collection

The quality of our final prototype for a position discovery service would be dependent on the performance of the underlying classifier. And the classifier would be the result of the training data that it was generated with. The systematic steps that were taken to obtain the training data are elaborated in the following.

2.1.1 Selection of Sensors

To exploit the capabilities of the sensors for our declared objective effectively, it seemed reasonable to explore what sensors are provided by the Nexus device to then decide which of them could be considered suitable to help determine the exact phone position.

In general, the Android platform supports three broad categories of sensors: motion sensors, environmental sensors and position sensors (Android Developer (1)). Motion sensors are useful for monitoring device movement and include accelerometer, gyroscope, gravity sensor and rotational vector sensor. Environmental sensors measure environmental parameters, such as ambient air temperature, pressure, illumination and humidity. They include light sensor, thermometer and barometer. Position sensors are capable of measuring the physical position of a device and include the magnetometer and the proximity sensor.

Our task required finding the phone position relative to the device's frame of reference as opposed to the world's frame of reference which would refer to the mostly outdoor geographical position of the phone (Android Developer (2)). From our literature study we learnt that this type of position, also referred to as the *phone sensing context* by Miluzzo et al. (2010) could not be captured by the GPS-sensor (Alanezi, Mishra, 2013) which has an accuracy of several meters and therefore would be too low for the type of position that we aimed to determine. The phone position in the device's frame of reference could hence be only implicitly inferred from the physical context of the user. For this reason, we placed our attention to the motion sensors which are sensitive to subtle device movements, such as tilt, shake, rotation or swing. Of all motion sensors, only the accelerometer and the gyroscope are hardware-based whereas the remaining sensors are software-based and as such combinations of the aforementioned sensors (Android Developer (1)). Based on the insights gained from our literature review we chose the accelerometer and the gyroscope as independent and hardware-based sensors

to measure device movement. Both return multi-dimensional arrays as sensor values. For the purpose of a more direct comparison as well as for visualization, which would be employed at a later stage, we calculated the magnitude from the three coordinates as the following scalar $\sqrt{x^2 + y^2 + z^2}$. As to the position sensors, we chose the proximity sensor which only indicates if there is an object in a distance of around 5 cm from the phone or not. And finally, from the environmental sensors, the light sensor was the only sensor that we found appropriate for our context. It would help to detect if the phone is in a covered position, like inside a bag or pocket.

An important parameter that we needed to decide upon was the rate at which sensor readings would be displayed. We found that Android provides three different modes for the sensor rates, which are as follows, SENSOR_DELAY_NORMAL, SENSOR_DELAY_UI, and SENSOR_DELAY_FASTEST. However, we noticed that the actual frequency within the same mode significantly varied across the different sensors that we used. Thus, we conducted several tests to ensure we set the mode for each sensor in a way that that we do not have duplicates for any of the sensor values in subsequent readings. One strategy to avoid this was to output the sensor values for all sensors relative to a sensor which had a slightly slower rate than the remaining sensors.

2.1.2 Selection of Positions

After choosing the set of sensors we had to make considerations about the specific data and its format we wanted to obtain from the data collection part and which would eventually result in our training data. An essential part of the training data was the target label for the phone positions that we would have to enter before collecting the sensor readings. To find an appropriate set of phone positions we employed both brainstorming and literature study. Finally, we chose 9 candidates for realistic phone positions. Those included shoulder bag, backpack, jacket pocket, front pants pocket, hip pocket and the user's hand while walking. All of the positions listed where captured while the user was walking at a normal speed, preferably on straight and even roads. By the user's hand which we labeled as "hand-motion" we referred to a context where the user would hold their phone in their hand while walking and without interacting with the phone. Apart from these positions we decided to include 3 more positions with an idle context of use. Those idle positions comprised placing the phone on the table, in a drawer, and holding it in one hand to interact with it while standing or sitting, e.g. to read messages, scroll up and down in a text but without any vigorous motions being involved such as rotating or tilting the device as would be required in a game for example. To clearly distinguish this hand position from the hand-motion position defined previously, we labeled it as "hand-still". Figure 1 shows snapshots of two views of the application for the sensor data collection.

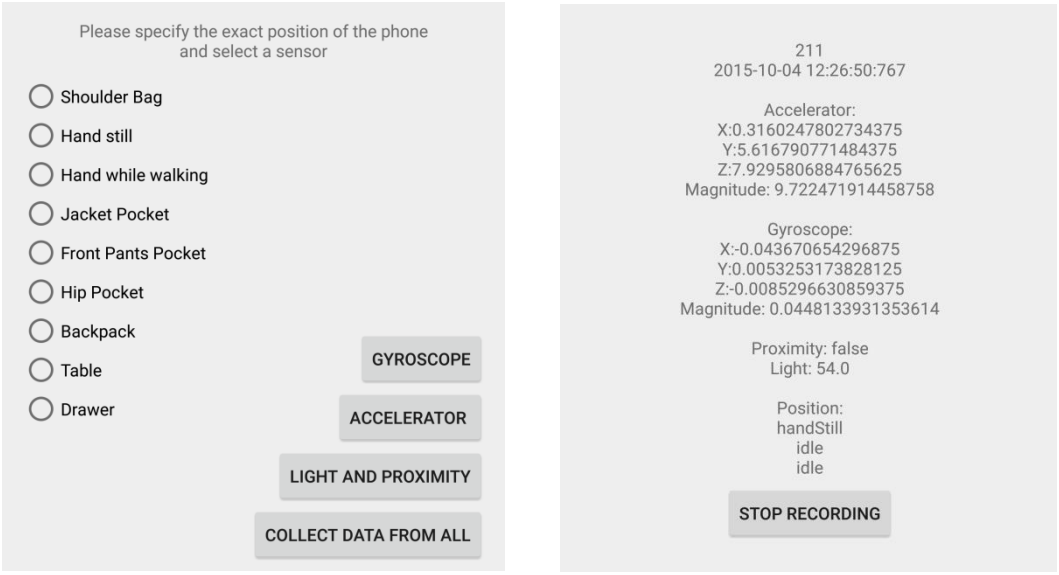

Figure 1: left: main view of the sensor data collection application for obtaining training data, right: view of the activity for data collection from all the sensors.

We were aware that depending on the quality of the collected data, a total of nine positions could be difficult to be distinguished by a classifier that we aimed to generate in the next step. With this in mind, we included two further columns as target labels that would generalize the exact phone position in the first class column. For example, we added the more general position “bag” for shoulder bag and backpack, and class “idle” for table, drawer and hand-still. For the pocket positions we divided the positions into lower and upper body since our findings from the preceding literature review suggested that the upper body was exposed to a lower extent of vibrations than the lower body (Alanezi, Mishra, 2013). The third most general position would simply attribute the class “pocket” to all of the pocket positions.

Finally, a record in our data set in a CSV format would consist of the ID or count of the current reading, the timestamp, the sensor values including the derived magnitude for the accelerometer and gyroscope, and the 3 target labels of which the first would denote the exact position and the remaining two the generalized positions. For all of the positions we collected a number of samples by the same user at different times (during day time) and at different frequencies of sensor readings.

2.2 Learning Features

After collecting a sufficient number of samples for each of the positions at different sensor rates, we found that it would be reasonable to visualize them before processing them to generate our classifier. Visualization would give us an insight as to which sensor rates should be selected, and as to which positions would be significantly distinguishable. The findings were supported by the use of statistical measures to gradually reduce the number of initially proposed positions. We used the tool DBPlot[1]

[1] DBPlot.2.4.8. March 2014.https://sourceforge.net/projects/dbplot/.

to visualize the samples for each of the positions and the machine learning tool Weka[2] (Hall et al., 2009) to extract some statistical features from our data sets. An important aim of the visualization and analysis was not only to examine whether a data set for a position exhibited a clear pattern but whether such patterns across different samples from a user with regard to the same position were consistent. Only if the collected data set and thus the resulting pattern for a position was not subject to randomness or significant irregularities, we would consider it for our classifier.

Once the set of positions was determined, we randomly chose multiple samples for the respective positions. We ensured that the distribution of our training data would be equal for all the positions and therefore we considered the same number and size of samples which were to be aggregated in a single CSV file. Despite the preceding analysis where we had ruled out some of the positions and sensor rates we now had again room for experimenting with different combinations of positions for the training data. The machine learning tool Weka accepts our aggregated CSV file with known position labels as input, and outputs a classifier which is able to classify the position for unseen input data once having learnt the underlying features. It is notable that Weka also outputs a thorough statistical analysis of the performance of the classifier and allows evaluations with test data, such as the 10-fold cross-validation test where the available data is split into 10 parts where 1 part is used as the test data while the remaining 9 parts are used as training data. To derive an averaged estimation of the classification error the described process is repeated 10 times such that none of the parts is used twice as test data. The statistical evaluations would help us to choose a classifier with the best performance.

3 Experiments

In the following the systematic methodology elaborated under 2 resulting from literature review and brainstorming is applied to the experimental setting involving the specified tools as well as the java-based applications for the Nexus 5 developed for this work. The different parameters and leverages for control are again emphasized at each stage such that they can be altered towards an optimal quality of the outcome represented by a classifier for positions.

3.1 Training Data

We collected data at two different sensor rates and output the sensor values for all sensors relative to the gyroscope sensor, since we found that for all of the modes for sensor rates the gyroscope was slower than the accelerometer and by printing the readings relative to the slowest sensor we ensured that we did not get any duplicates in subsequent sensor readings. The higher frequency we used was around 250 Hz (readings per second – corresponding to mode DELAY_FASTEST for gyroscope sensor) and the lower frequency was around 5 Hz (corresponding to mode DELAY_NORMAL for gyroscope sensor). After rendering several plots with different attributes for the y-axis, we decided it was sufficient to plot only the magnitudes for the gyroscope and the accelerometer. Since the values for proximity and light were rather

[2] Weka.3.4. June 2011.https://sourceforge.net/projects/weka/.

consistent over time we treated them separately and did not consider them in our visualization.

The visualization of random samples collected at two different frequencies gave us notable differences as can be seen in figure 2 and 3. The plots for the data collection at the lower sensor rate did not resemble any pattern whatsoever since the data points for the magnitudes were randomly scattered in a bounded interval of the y-axis for both the gyroscope and the accelerometer. On the contrary, the data samples collected at a higher frequency, rendered clear and mostly smooth patterns for all of the nine positions that were considered initially.

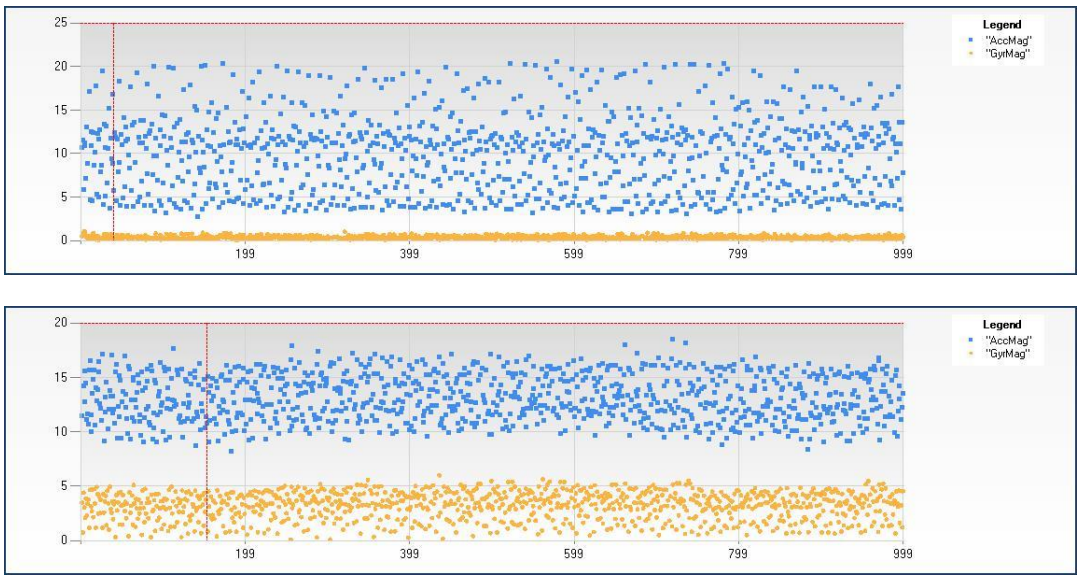

Figure 2: Magnitudes of accelerometer and gyroscope at 5 Hz; upper: backpack, lower: hand-motion.

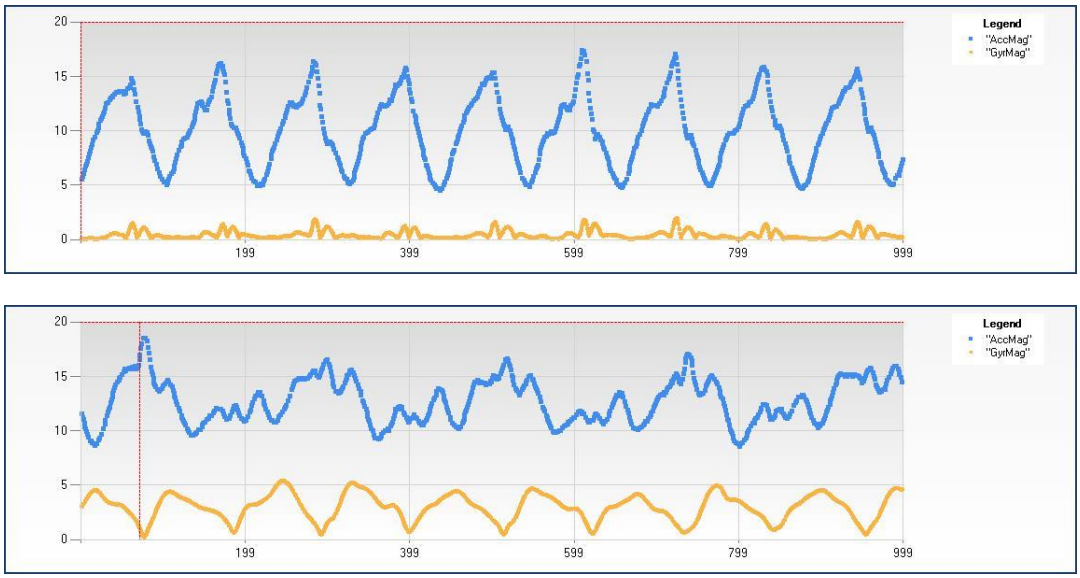

Figure 3: Magnitudes of accelerometer and gyroscope at 250 Hz; upper: backpack, lower: hand-motion.

Based on these findings, we considered only samples collected at the higher sensor frequency for determining the positions that our final application should support. Furthermore, we observed that all of the randomly chosen samples for the same positions mostly rendered consistent patterns. What we primarily took into account for the selection of the final positions, was the distinctness and the smoothness of the visualized pattern for a position. Besides, simpler patterns without many oscillations were preferred. As the preceding literature study revealed to us, the patterns for positions from the lower body reflected more oscillations than the upper body (Alanezi, Mishra, 2013). Figure 4 shows the plots for the magnitudes for all 3 pocket positions. Although backpack and shoulder bag were not generalized to upper body in our alternative class labels, we found that the plots for the front pocket, backpack and shoulder bag exhibited some similarities in that they rendered graphs with smoother slopes and both smaller and simpler cycles with mostly a lesser number of local peaks within a cycle. This could be explained by the fact that all 3 positions were closer to the upper body than the lower body. Furthermore, we observed that hand-motion was a position that rendered a very smooth graph for the magnitudes of both accelerometer and gyroscope. It was the only position where the magnitudes for the gyroscope had clear oscillations with significantly higher amplitudes than in the case of the plots for the other positions (see lower part of figure 3). This was consistently confirmed by a higher standard deviation of the magnitude of the gyroscope for the hand-motion position. While hand-motion reached a standard deviation of around 1.2 rad/s, the rest of the positions reached a gyroscope magnitude of under 0.6 rad/s as we could extract from the statistical analysis in Weka.

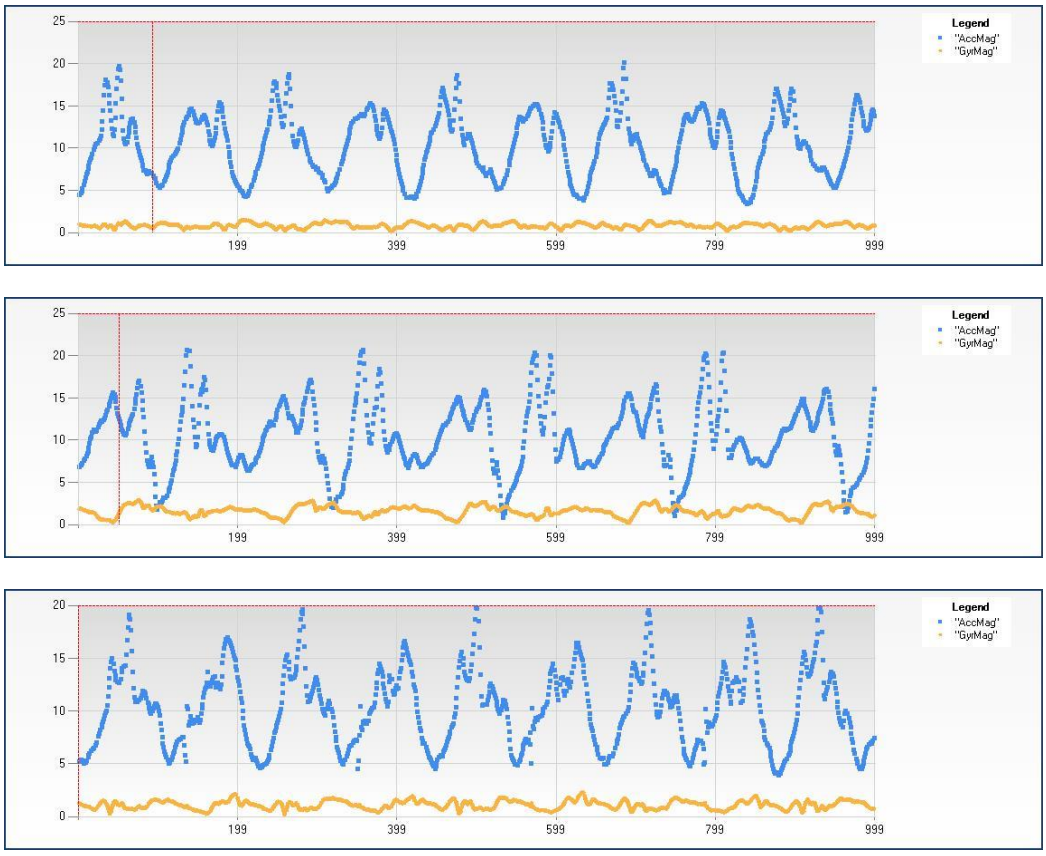

Figure 4: Magnitudes of accelerometer and gyroscope at 250 Hz of all 3 pocket positions in the following order: jacket pocket (upper body), front pants pocket, hip pocket (both lower body).

3.2 Classifier Setup

Taking all observations into account, we finally agreed on a set of positions including one upper body position, backpack, one lower body position, hip pocket, and hand-motion which exhibited significantly large amplitudes for the gyroscope magnitude. From the idle positions we only included the table position, since the difference between table and drawer was, as expected, merely in the value of the light sensor which could be easily handled in our final code. As far as the "hand-still" position was concerned, it turned out to be non-different from the table position, despite the fact that the phone was held in a tilted way, as is natural when a user interacts with it in an idle physical context. Therefore we excluded the position hand-still from any further considerations.

For the classifier we chose a training data set with two randomly selected samples of 18,000 data points each and aggregated the samples for all the positions we previously agreed upon. We noticed that Weka offered a number of different classification algorithms but most of them were merely restricted to the output of the analysis without the generation of a classifier. In the literature review we found that for classification problems involving phone positions function as the Naïve Bayes, Simple Logistic Regression or decision tree algorithms such as the J48 were used. We mainly worked with the J48 algorithm since it generated classifiers that scored particularly well in the 10-fold cross-validation tests and furthermore did not yield too complex classifiers for our training data. Complex classifiers have the risk of overfitting attached to them, which implies that the classifier performs well on the particular training data but performs poorly on test data. On account of this consideration, we ensured to prefer a less complex model as far as possible.

3.3 Results and Discussion

For a comparative analysis, we generated 3 different classifiers based on 3 sets of training data all of which considered a different set of positions. Our preferred set of position labels was the one proposed before {backpack, hip pocket, hand-motion and table}. An alternative set of position labels was chosen as {jacket pocket, hip pocket, hand-motion}. And finally, we deliberately chose to generate a more complex model by including a higher number of positions based on the set {backpack, jacket pocket, front pocket, hip pocket, hand-motion, backpack, table}.

From the statistical evaluation that Weka generated alongside the actual classifier, we could infer that there was a correlation between the complexity of the classifier and the number of positions that we wanted our classifier to support. Furthermore, we learnt that the combination of positions should be chosen considerately to obtain a classifier with low complexity which would consequently have a higher accuracy of classification as overfitting would be avoided. Table 1 presents an overview of the complexity and performance of J48-classifiers that we obtained using the different combinations and number of position classes

Set of supported positions	Size of tree	Number of leaves	Accuracy of classification according to 10-fold cross validation test
{jacket pocket, hip pocket, hand-motion}	212	423	99.5458 %
{backpack, front pocket, jacket pocket, hip pocket, hand-motion, backpack, table}	243	485	99.7040 %
{backpack, hip pocket, hand-motion, table}	4	7	100 %

Table 1: complexity and performance of J48-classifiers for 3 different sets of positions

It is notable that the combination {backpack, hip pocket, hand-motion and table} which was our preferred one based on the visual evaluation of the patterns from the plots, generated a fairly simple tree of size seven with only four leaves, which is in fact the minimum number of leaves for a classifier with four target labels. The performance of this classifier reached 100% implying that all instances from the test data were classified correctly in the 10-fold cross-validation test. We would like to point out that despite the fact that the classification accuracy of the validation tests for the other two sets of positions was equally large, the risk of overfitting is likely to be higher, since the complexity of the models is high in both cases when compared to our best classifier. Table 2 shows the ROC-analysis of the best classifier as well as the confusion matrix for it. The confusion matrix indicates for all of the considered labels how many of the inputs from the true respective labels were classified as those. Our classifier led to perfect results since we have the full number of inputs on the main diagonal, and zeros elsewhere. Based on the confusion matrix Weka calculated the values for the ROC-analysis considering statistical ratios such as precision, true-positive rate, recall etc., which again lead to perfect values since for none of the classifications we have a deviation from the true labels which would push the values below the highest score of 1.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0	1	1	1	1	hand-motion
1	0	1	1	1	1	hip
1	0	1	1	1	1	table
1	0	1	1	1	1	backpack
1	**0**	**1**	**1**	**1**	**1**	**Weighted Avg**

=== Confusion Matrix ===

a	b	c	d	<-- classified as
18000	0	0	0	a = hand-motion
0	18000	0	0	b = hip
0	0	18000	0	c = table
0	0	0	18000	d = backpack

Table 2: ROC-analysis and confusion matrix for best classifier based on set {backpack, hip pocket, hand-motion and table}.

4 Use Case: Position Discovery Service

After selecting the classifier we were able to proceed with the implementation of our position discovery service for the Nexus smartphone. It consists of a mobile phone application and a desktop application. The phone application would again collect sensor data at the same modalities at which the training data was collected in the first application. The files created are then transmitted to a desktop application which would assume the actual classification and then display the current position of the phone on a user interface. In the following, the logic of both the applications is elaborated.

The phone application requires the collected data to be uploaded online, in order for the desktop application to access and process it. Therefore, the user has the possibility to link a Dropbox account, where the data will be uploaded in CSV files at a certain rate which can be modified on the main screen. The default rate is 5 seconds, implying that the data is collected and written into a buffer for a period of 5 seconds, at the end of which the buffer is flushed into a Dropbox file. We were not able to choose a higher rate, as this would have overwhelmed the upload queue, causing growing delays. Figure 5 shows a snapshot of the settings view of the smartphone application of the positions discovery service.

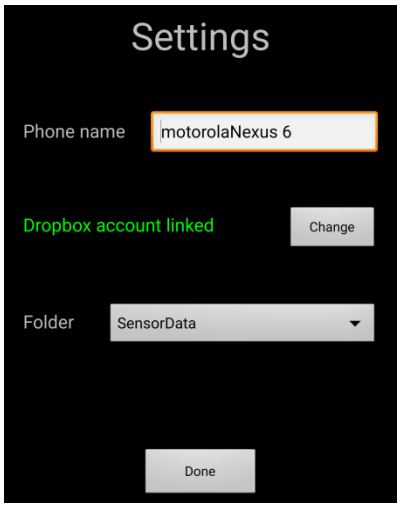

Figure 5: Settings view of the smartphone application of the position discovery service.

Once the settings regarding the phone name and Dropbox account are saved, the user may return to the main screen to start the data collection. The sensor data will be written into the collection buffer after a period of 15 seconds, so that the phone can be put in the desired position. The uploading process can be paused or relaunched by pressing the "Stop" and "Read" button, respectively.

The desktop application uses the algorithm generated by Weka in order to classify the sensor data collected with the smartphone application. When the program is launched, the user is asked to link his Dropbox account, in order for the collected data to be downloaded and processed. Once this is successfully completely, the application starts scanning the folder where the files were uploaded using the mobile App, searching for changes like newly created or modified files. When a change is detected, the file is downloaded and the containing data starts being processed.

The first change that is detected will trigger launching a window that displays a table with the following columns: the name of the phone that uploaded the data (which can be found by parsing the file's name), the current position of the phone and a timestamp indicating when this position was last updated. The application is entirely scalable as other phones' positions can be added to the table and viewed in real time as the data is being read from the online folder. Figure 6 shows a snapshot of the user interface of the desktop application.

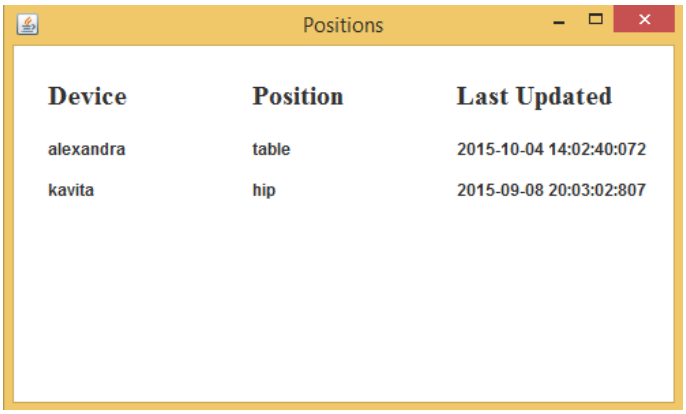

Figure 6: User interface of the desktop application displaying current phone positions.

5 Conclusion

Our declared aim was to develop a prototype for a position discovery application for the smartphone which could possibly add value to context-aware applications and services. We approached this challenge with the Android-powered Nexus 5 smartphone. The position discovery problem involved finding the exact position of the phone in the device's frame of reference. Since no such sensor, as for instance, the GPS sensor, is capable to detect the exact phone position in a static context, we had to essentially rely on motion sensors which measure subtle movements of the device at remarkably higher frequencies. On account of this, nearly all positions we considered initially required the user to walk when collecting the sensor data. The data samples collected at the higher frequency rendered clear patterns at the stage of visualization. The plots for the lower body positions exhibited more oscillations during longer cycles whereas the patterns for the upper body positions showed smoother slopes as well as shorter and simpler cycles and a lesser number of local peaks. The criteria for selecting the positions that we wanted our final application to support included a low complexity of the pattern and a possibly high distinguishability when compared to other positions. Furthermore, we noticed that there was a correlation between the number of positions that we considered and the complexity of the model which was generated by Weka using a set of samples for different positions. Hence, a third criterion was to reduce the number of positions if the complexity of the model that we obtained was regarded as fairly high. The danger of selecting a complex model for our classifier involved the issue of overfitting which implies small training error but higher validation error on unseen input. Due to these considerations we chose the following set of position classes {backpack, hip pocket, hand-motion and table}. The training data based on these positions generated a classifier which as a J48-decision-tree had a size of 7 and comprised 4 leaves, which is in fact the minimum number of leaves for a classifier with

4 classes. Our preferred classifier achieved a 100% classification accuracy in the 10-fold cross-validation test and optimal values in the ROC-analysis. After successfully implementing the generated classifier in our final application for the Nexus device, we developed a desktop application with a user interface where the current position of the phone would be displayed. Additionally, for each session of the position discovery application the output was sent to a CSV file which was uploaded on Dropbox.

Limitations and Future Work

We were aware that the quality of the training data would be determined by the consistency of the data with regard to a position. We ensured this consistency by analyzing the visualized patterns which indeed exhibited consistent patterns across different samples for the same position. However, it should be noted, that the training data was collected from one user only. We presume that patterns from different users could be similar since for example our results also confirmed the findings from existing studies in the point that the lower body positions were exposed to more vibrations which were reflected in more oscillations in the corresponding patterns. However, the cycles could be of different sizes depending on the user's natural speed of walking. And the amplitudes could vary due to the force of the rhythm of the user's movement. Although we did not investigate further into variations resulting from training with different users, we presume that including samples from different users might have rendered a more complex model for the same set of positions that we had eventually selected. On the other hand, this limitation can also be seen as a strength of our prototype. Since the resulting pattern for any of the positions would be subject to the user's individual physical constitution, it could be a reasonable approach to integrate both training and generating the classifier into one application. Such a personalized application would be capable of learning the user's physical motions from which the smartphone position could be inferred based on the personalized classifier.

Lastly, we would like to point out that in our analysis and implementation we did not consider the potential of acoustic data from the built-in microphone sensor. However, there exist a number of researches employing audio signal processing to detect patterns in the environmental noise in order to infer hints on the position of the smartphone, e.g. compare Diaconita (2014) or Miluzzo et al. (2010).

Acknowledgements

I would like to thank Prof. Delphine Reinhardt from Computer Science Institute 4 at the University of Bonn for her supervision of the lab that this work resulted from. I also thank the fellow student Alexandra Similea who, after joining me at a late stage of the lab not only shared her research experience with me but also contributed to the development of the second mobile app as well as the desktop application for the use case under 4.

References

Alanezi, K., Mishra, S. (2013) Impact of Smartphone Position on Sensor Values and Context Discovery. Computer Science Technical Reports. Paper 1030.

Android Developer (1), Sensors Overview. http://developer.android.com/guide/topics/sensors/sensors_overview.html, accessed 2015-03-10.

Android Developer (2), Position Sensors. http://developer.android.com/guide/topics/sensors/sensors_position.html, accessed 2015-03-10.

Diaconita, I., Reinhardt, A. Englert, F., Christin, D., Steinmetz, R. (2014) Do you hear what I hear? Using acoustic probing to detect smartphone locations. Pervasive Computing and Communication Workshops (PERCOM Workshops), 2014 IEEE International Conference, Budapest, Hungary, pp. 1-9.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutmann, P., Witten, I., H. (2009) The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11, Issue 1.

Miluzzo, E., Papandrea, M., Lane, N. D., Lu, H., Campbell, A. T. (2010) Pocket, Bag, Hand, etc. - Automatically Detecting Phone Context through Discovery. First International Workshop on Sensing for App Phones (PhoneSense) at SenSys'10.