Anomaly Detection for Mobile Device Comfort

Mehmet Bicakci, Babak Esfandiari, Stephen Marsh

Anomaly Detection for Mobile Device Comfort

Mehmet Vefa Bicakci[1], Babak Esfandiari[1], and Stephen Marsh[2]

[1] Department of Systems and Computer Engineering,
Carleton University, Ottawa, Ontario, Canada
mehmetvefabicakci@cmail.carleton.ca
babak@sce.carleton.ca
[2] Faculty of Business and Information Technology,
University of Ontario Institute of Technology,
Oshawa, Ontario, Canada
stephen.marsh@uoit.ca

Abstract. As part of the Device Comfort paradigm, we envision a mobile device which, armed with the information made available by its sensors, is able to recognize whether it is being used by its owner or whether its owner is using the mobile device in an "unusual" manner. To this end, we conjecture that the use of a mobile device follows diurnal patterns and introduce a method for the detection of such anomalies in the use of a mobile device. We evaluate the accuracy of our method with two publicly available data sets and show its feasibility on two mobile devices.

Keywords: Anomaly Detection, Device Comfort, Mobile Device, Soft Security

1 Introduction

Mobile devices (such as smartphones and tablets) have become popular convergent platforms that can be used for many tasks from banking to photography to e-mail. Modern mobile devices also come with a large number of sensors including accelerometers, gyroscopes, magnetometers, proximity and ambient light sensors, Global Positioning System (GPS) sensors, and Bluetooth, WiFi and cellular connectivity.

The data that can be garnered from the aforementioned sensors and the nature of the task that the user is carrying out on the mobile device can be used by the mobile device to have a pretty accurate picture of the contextual and behavioural patterns of the mobile device user.

We envision a mobile device that can get to know its owner. Such a mobile device would be able to detect if it is being used by a user other than its owner or whether the owner is behaving in an "unusual" manner based on the behavioural and contextual patterns that the owner established with the mobile device. We believe that the detection of such "anomalies" in the context and user behaviour are valuable for protecting the mobile device, the data on the mobile device, and last but not least, the owner of the mobile device.

Device Comfort [1] is an application of computational trust to (*soft*) mobile device security aiming to provide the aforementioned protection and more by making use of behavioural biometrics, contextual information and policy to let the mobile device reason about its owner's behaviour and the context. Via its contextually determined security posture, or "comfort level," and the policy elements, the mobile device can warn its owner against performing potentially dangerous tasks and, if necessary according to the policy, prevent such tasks from being performed. As a result, Device Comfort aims to help the user understand and reflect upon the potentially harmful consequences of the (possibly unusual) behaviours he/she performs with a comfort-enabled computing device.

As the last two paragraphs hint, the *threat model* considered by Device Comfort (and hence this paper) is related to soft security and human aspects of computing. As part of its threat model, Device Comfort aims to defend a personal computing device (and its user) against "unusual" and/or "inappropriate" use of the device, the definition of which is an area of research. User behaviour resulting from distractions or inattention, which has the potential to compromise security, is also considered as a possible threat. Last but not least, the Device Comfort threat model also considers physical intrusions and theft.

We believe anomaly detection is one of the building blocks of Device Comfort, where an "anomaly score" can be one of the sources of information that are used to determine the comfort level of a mobile device. As such, in this work we focus on performing anomaly detection using the behavioural and contextual patterns that a mobile device user establishes with his/her mobile device for the enablement of Device Comfort.[3]

We conjecture that there exists a 24-hour cycle in the behavioural and contextual information that can be sensed via the sensors and the operating system of a mobile device, and we propose to exploit such diurnal patterns for anomaly detection purposes. Our approach consists of partitioning each day's data into *time slices* that have fixed and equal length and comparing the time slices of "today" to those of the past days.

We evaluated the accuracy of our approach using two data sets containing mobile device usage data, and to show the feasibility of our approach on actual mobile devices, we deployed our software on two mobile devices.

We find that with the first data set cellular location, phone call, and Bluetooth discovery features contribute more to the accuracy of our method compared to text message contacts and the names of the applications started by the user. With the second data set, we find that the called phone numbers and Bluetooth discovery results make a greater contribution to overall accuracy compared to WiFi discovery results and text message contacts.

As part of investigating the feasibility of deployment, we find that performing anomaly detection on actual mobile devices is feasible even with relatively aggressive anomaly detection parameters, where the computational performance

[3] In particular, the policy elements which determine *what* happens when an anomaly is detected are left for future work.

is not affected from the user's point of view, whereas the battery life is affected negatively.

The rest of this paper is organized as follows: In the next section, we review work related to anomaly detection on (mobile and non-mobile) personal computing devices, the section following which introduces and describes our methodology. Afterwards, we describe our evaluation strategy and present our results. The paper ends with a concluding section which includes a number of future work items as well.

2 Related Work

Anomaly detection for mobile (and non-mobile) personal computing devices is an area in which numerous proposals have been made. We summarize our review of related work in four subsections, the first of which briefly discusses the algorithms and features used by the authors, followed by a subsection noting the data sets used by the authors. The third subsection reports on the deployment of the related proposals. In the fourth subsection, we briefly compare our approach to those of the related proposals.

2.1 Methodologies and Used Features

Shi et al. [2] utilize a mobile phone user's behavioural patterns in the form of GPS-based location, phone call, text message exchange and web browsing history. The user's location is spatio-temporally clustered using the Gaussian Mixture Model clustering algorithm. For each feature other than the location, probabilistic models conditioned on the time of day are built, where, given the time of day and the number of hours since the last "good" (i.e., "observed before") event and the number of "bad" events in the past 24 hours, an authentication score is computed. The feature values specific to certain times of the day are not taken advantage of. For example, the approach does not consider whether the *specific* phone number being called is usually called in the morning or the afternoon, but only considers that it is a "good" phone number that had been called before.

Li [3] uses the names of started applications, location (as inferred from cellular towers), and phone call and text messaging histories for anomaly detection, which is performed by considering how prevalent feature values fused with the cellular location are in the training data. Li does not consider the time of day as a feature, noting in [3] that it contributes negatively to the overall performance.

Yazji et al. in [4,5] propose to detect anomalies in the spatio-temporal patterns of a mobile phone user via two methods. The first method involves the summarization of the distribution of user's presence to produce a spatio-temporal matrix, whereas the second method exploits the Markov properties of trajectories. Both methods are very specific to spatio-temporal analysis.

Branscomb [6] investigates whether the number of seconds of a mobile phone user's time spent in each application category can be used to verify the identity of

the user. Temporal patterns are modelled via a binary feature reflecting whether the applications are being used on a weekday or during a weekend.

Crawford [7] verifies the identity of the mobile phone user with keystroke and voice dynamics, the feature vectors describing which are classified using Naive Bayes, Decision Tree and k-Nearest Neighbours classifiers. Zhu et al. [8] propose to classify the motions of the mobile phone user (as measured with accelerometers and gyroscopes of a mobile phone) using k-Means clustering and an n-gram language model. Crawford and Zhu et al. do not consider the time of day as a feature, and it may not be sensible to do so as the used features may not vary according to the time of the day.

In [9], Yazji et al. target laptops and perform anomaly detection via the use of k-Means clustering, where the timestamped file-system and network accesses made by the user in every five-minute-long time quanta are classified as normal or anomalous.

In contrast, in [10] Salem et al. propose to detect "masquerader" attackers by focussing on their search-related file-system access patterns on desktop computers, where accesses made at every two-minute-long time quanta are summarized and input to the one-class SVM algorithm for classification. The authors do not use the time of day as a feature.

2.2 Data Sets

We observe that only Li [3] and Yazji et al. (in [4,5]) use data sets available to the research community. Li uses the Reality Mining data set [11], and Yazji et al. use the Reality Mining and GeoLife [12] data sets.

To the best of our knowledge, the *RUU* (*Are You You?*) data set collected by Salem et al. for the evaluation of their proposal in [10] had been published in the past, but is no longer available as of this writing.

2.3 Deployment

Mobile devices have limited computational power and battery life. As a result, we believe that it is important to verify the deployment feasibility of an anomaly detection method targeted for mobile devices.

In the three proposals by Yazji et al., a server works hand-in-hand with the mobile device to perform computation- and energy-intensive tasks. This is in contrast to the other proposals we have reviewed, which perform anomaly detection locally on the mobile device.

We notice that only Zhu et al. and Branscomb have fully deployable solutions which implement data collection and anomaly detection functionalities.[4] Crawford, Salem et al., Shi et al., and Yazji et al. (in [9]) only deploy the data collection logic, whereas Li and Yazji et al. (in [4,5]) do not have a deployable implementation.

[4] We should note that Zhu et al. evaluate their approach in a deployed setting indicating the completeness of their implementation. According to [6, p. 30], Branscomb has a "proof-of-concept app."

2.4 Discussion

Our approach, as will be introduced in the next section, makes use of a time-quantum-based summarization method similar to some of the related proposals, and we use the time of day as a feature while taking into account the feature values that are specific to certain times of the day.

In contrast to most related proposals, we evaluate our method using *two* data sets that are available to the research community – the Reality Mining [11] and the Social Evolution [13] data sets.

Finally, we deploy our approach on two mobile devices to determine the feasibility of performing anomaly detection on an actual mobile platform.

3 Methodology

We envision that a comfort-enabled mobile device would retrain itself every midnight based on the past N days' behavioural and contextual data, and classify the data encountered on the day following the midnight as anomalous or normal based on the training data.

According to Chandola et al. [14], this scheme corresponds to a semi-supervised anomaly detection approach, where a one-class machine learning algorithm builds a model based on only the normal data instances and is tested against normal and anomalous data instances.

3.1 Data Model

To model data, we use a "summarization" method, where we divide each day into equal- and fixed-length *time slices* of configurable size. For example, if we choose four-hour-long time slices, then we will have six time slices per day of mobile device usage, each of which summarizes the behavioural and contextual patterns in the use of a mobile device.

Our summarization method consists of instantiating per time slice a data structure, each of which contains the following pieces of information: (1) Time of day at which the time slice begins, (2) time of day at which the time slice ends, and (3) a set of hash tables, one per feature type that is being summarized.

Each of the aforementioned hash tables is populated as follows: the hash table is keyed with the feature value, and the keys of the hash table point to values indicating how many times the feature value in question had been observed in the time slice to which the hash table belongs.[5]

For example, for a hash table for the text message exchange feature, the hash table would be keyed with tuples in the following form "$(phoneNumber, direction)$", and the hash table values corresponding to the keys would be the *number of times* for which text messages had been exchanged with the phone number in the given direction – incoming or outgoing.

[5] For the cellular location (cellular area and cell identifier) feature, we record the number of minutes spent in the particular location instead.

Using such a data model, we aim to find diurnal patterns in a mobile device user's behaviour, where if the time slice ts belonging to "today" does not match any time slices belonging to the past N days with room for some temporal error, then we can consider today's time slice ts to be anomalous – the mobile phone may have been compromised physically, or the user may have been behaving "unusually."

3.2 Anomaly Detection Method

To detect anomalies, we use a variant of the k-Nearest Neighbours algorithm adapted to anomaly detection. The use of a distance-based anomaly detection algorithm allows us to use custom distance functions with complex data structures.

The overall strategy of the k-Nearest Neighbours-based anomaly detection algorithm we use was introduced by Eskin et al. in [15]. The algorithm consists of finding in the training data set the k nearest neighbours of each test data instance d_{test}, and for each d_{test}, summing the distances of d_{test} to its nearest neighbours to obtain an anomaly score. The anomaly score of each test data instance d_{test} is then compared to an operator-set threshold value to make a prediction: if a distance sum is greater than the threshold value, then anomalous, otherwise, normal. Using this logic, only data points with very far neighbours or not a lot of near neighbours are predicted as anomalous by the algorithm.

Distance Function To compute the distance between two time slices we use the distance function in the following equation:

$$distance_{timeSlice}(ts_1, ts_2) = \begin{cases} MAX\_DISTANCE & \text{if } |t^{ts_1} - t^{ts_2}| > DELTA \\ \dfrac{\sum\limits_{i \in FNE} distance_f(f_i^{ts_1}, f_i^{ts_2})}{|FNE|} & \text{otherwise} \end{cases} \tag{1}$$

where t^{ts_x} represents the time of day at which time slice ts_x begins, and $f_i^{ts_x}$ represents the ith feature of time slice ts_x, and FNE represents the set of feature types which are non-empty in both time slices.

The constant $DELTA$ allows us to configure the "leniency" in the finding of diurnal patterns with respect to the time of day. As can be seen in Equation 1, if the two time slices ts_1 and ts_2 start at times of the day that are more than $DELTA$ hours apart, we assign $MAX\_DISTANCE$ as the distance between the two time slices in order to never consider these two time slices as near neighbours.

If ts_1 and ts_2 start at relatively similar times of the day according to the $DELTA$ constant, then we use the $distance_f$ function to compute the pairwise distance of each feature (hash table) in the two time slices, with the restriction

that we only consider the feature types which are not empty in both time slices. The average of the computed distances is assigned as the distance between the two time slices in question, which corresponds to equally weighting each feature type.[6]

Distances between Features The distances between the values of each feature type are obtained using the Jaccard distance metric, which is implemented in the function named $distance_f$ referenced in Equation 1.

To compute the Jaccard distance, we "convert" the feature hash tables into sets by considering only the keys in the hash tables, which correspond to, for example, the phone numbers which had been called in the time slice to which the feature belongs. Afterwards, the Jaccard distance is computed as follows: $1 - \frac{|A \cap B|}{|A \cup B|}$, where A and B correspond to the sets obtained by hash table conversion process.

With the Jaccard distance we ignore the values in the feature hash tables which indicate the number of times each feature value had been observed in a time slice. We have also experimented with the Binary Weighted Cosine (BWC) distance [16] which takes into account the number of observations by making use of the cosine similarity. In our experiments, the BWC distance produced results that are slightly worse than those obtained with the Jaccard distance. This phenomenon can be explained with the fact that the BWC distance makes use of cosine similarity in addition to Jaccard similarity, the former of which considers two feature vectors (i.e. the features in two time slices) similar based on their relative orientations in Euclidean space.[7] We conjecture that in our experiments the number of times where taking into account frequencies of feature values improves the classification results is less than the number of times where taking frequencies into account degrades the results. As a result of the above reasoning we use the Jaccard distance in this paper.

Empty Feature Values and Empty Time Slices One open problem is the handling of features for which values are not available in both time slices, which we call "empty feature values." While one can consider two empty hash tables as "equal" with a distance of zero, we choose to ignore feature types corresponding to empty hash tables in both time slices based on the empirical observation that doing so enables us to obtain more accurate results.

Another open problem is the handling of "empty time slices," which occur when the mobile device is switched on but no behavioural/contextual data is available – i.e. no phone calls are made, no location updates are observed, and no Bluetooth or WiFi discovery results are available.

[6] We should note that a custom weighted sum scheme is certainly possible, but is left for future work.

[7] For example, two time slices in which two phone numbers are called with the same *ratio* of frequencies would be parallel in Euclidean space, and therefore very similar according to cosine similarity.

While the existence of an empty time slice at a time of the day which is usually very "busy" in terms of the collected features may indicate an anomaly, we choose to ignore empty time slices because we cannot reliably verify the identity of a mobile device user if there is no data available.[8]

4 Evaluation

4.1 Data Sets

We evaluate our approach using the Reality Mining and the Social Evolution data sets.

The Reality Mining data set [11] is the result of a study carried out in the 2004–2005 academic year, and includes data belonging to roughly 100 participants – faculty, staff and students of MIT, where for each participant the names of started applications, Bluetooth discovery results, location (as inferred from cellular towers), and phone call and text message exchange histories were recorded along with timestamps.

The Social Evolution data set [13] was collected during the 2008–2009 academic year and includes the features collected from the mobile phones of roughly 80 participants in a dormitory in MIT: Bluetooth and WiFi discovery results, and phone call and text message histories.

The Reality Mining data set was used by Li [3] and Yazji et al. [4,5] for evaluation purposes as well.

4.2 Feature Extraction

From the Reality Mining data set, we extract the following features: timestamps, names of applications started, the results of Bluetooth discoveries, from which we extract the name and Bluetooth address of the discovered devices, cellular tower information (cellular tower area and cell identifiers), and the numbers with which phone calls and text messages were exchanged. For text messaging, we extract the direction (incoming vs. outgoing) of text messages as well.

From the Social Evolution data set, we extract the following features: timestamps, the results of Bluetooth discoveries, where we use the identifier of the detected study participant as the "fake" Bluetooth address and name of a detected device,[9] the results of WiFi discoveries, and phone call and text message exchange histories.

[8] Furthermore, the existence of empty time slices may indicate the need for more features in order to discriminate the empty time slices from others.

[9] We use fake Bluetooth addresses and names, because only the identifiers of the *participants* of the study were detected and recorded in Bluetooth discovery results in the Social Evolution data set.

4.3 Evaluation Method

The data sets that we use for evaluation lack anomalous mobile device usage data. As a result of this limitation, we use a "1-versus-rest" evaluation scheme, where for each data set, one at a time, we consider each participant of the data set as the user/owner of a mobile device, and consider all other participants as "attackers" or, in other words, sources of anomalous usage data. While this evaluation scheme may not be realistic in the simulation of anomalous usage, it has been used in a large number of related proposals: [2,3,4,5,6,7].

To evaluate our method, we use the following strategy: We instantiate time slices from the data of all of the data set participants, and we consider the data belonging to the current mobile device "owner" data set participant $User_A$ as a stream of days composed of *normal* time slices. We start by training an anomaly detection model on the first D days' time slices belonging to $User_A$. Afterwards, we use the time slices on $(D+1)$th day of $User_A$'s data as *normal* testing data, via which we obtain true negatives and false positives. Finally, as *anomalous* testing data, we take a sample of the time slices belonging to the *other* participants of the data set $User_x$, where $x \in \{B, C, D, E, ...\}$. These anomalous time slices let us obtain true positives and false negatives.[10]

Once the testing for this evaluation "iteration" is complete, we shift the training period to the right by one day so that the training period is composed of the time slices from the days 2 to $(D+1)$ belonging to $User_A$. The time slices on day $(D+2)$ are considered as *normal* time slices for testing, and another sample of time slices belonging to the other participants is taken and considered as *anomalous* testing data.

This procedure is continued until we reach the end of $User_A$ data "stream," after which we repeat the same procedure where we consider $User_B$ as the mobile device owner, and all other users (including $User_A$) as attackers/sources of anomalous usage.

4.4 Evaluation Metric

Because of the class imbalance inherent in our evaluation method, where anomalous time slices are more numerous than normal time slices, we use the Area Under Curve (AUC) summary metric as the performance evaluation metric. AUC is a measure of the correctness of the machine learning algorithm under evaluation, where an AUC value of 0.5 indicates an algorithm that cannot make predictions better than random guessing, whereas an AUC value of 1.0 corresponds to perfect prediction performance. As a result, the higher the AUC value, the better the performance of an algorithm. Unlike accuracy, the AUC metric is not affected by class imbalance.

In order to obtain an AUC value for one evaluation "iteration," we vary the anomaly score threshold to obtain all possible combinations of false positive

[10] We have verified that the manner in which we sample anomalous time slices does not introduce more than ± 2.5 AUC percentage points.

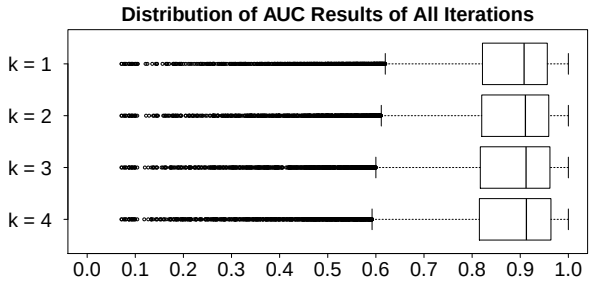


Fig. 1: Performance of k-NN with the Reality Mining Data Set (Each boxplot summarizes 12431 iterations.)

and true positive rates corresponding to the performance of the anomaly detection algorithm's performance. After plotting the true positive rate against the false positive rate, we compute the area under the resulting Receiver Operating Characteristic curve to obtain an AUC value for one evaluation "iteration."

4.5 Classification Performance

We present our results in the form of boxplots, which depict the variation of AUC values across all evaluation "iterations," each of which correspond to the performance of an anomaly detection model on a particular test day for a particular data set participant.

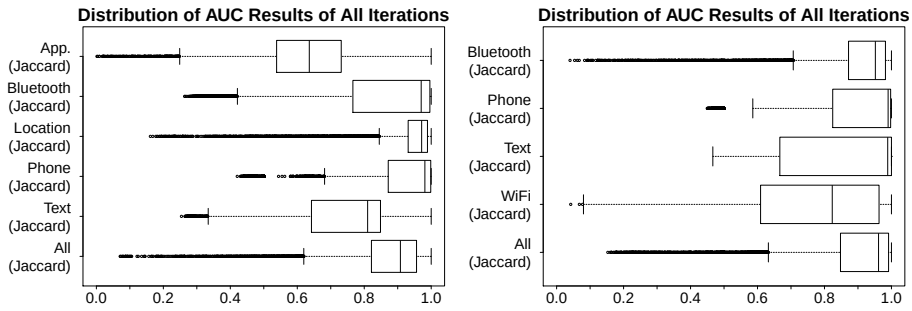
Parameter selection with k-NN With the aim of selecting the most appropriate value of k, we experiment with the k-Nearest Neighbours algorithm on the Reality Mining data.

In these experiments, we use all of the features extracted from the Reality Mining data set, and set the time slice length and leniency equal to 4 hours and set the training period length equal to 21 days.

Our results can be seen in Figure 1, from which one can see that varying k does not appear to affect the performance of k-NN. Based on these observations, we continue our experiments with k set to 1.

Results with Individual Features In this set of experiments, we enable each feature in each data set one by one to determine how much each feature contributes to the overall performance of our approach. We use k-NN with k set to 1, and we set the time slice length and leniency equal to 4 hours and set the training period length equal to 21 days.

We present our results in Figures 2a and 2b, from which we observe that with the Reality Mining data set the cellular location, phone call history and Bluetooth discovery features perform better than the other features, and with the Social Evolution data set the phone call history and Bluetooth discovery perform better than the text messaging history and WiFi discovery features.



(a) Reality Mining Data Set (Number of iterations: App.: 11472, Bluetooth: 10268, Location: 11247, Phone: 10498, Text: 6644, All: 12431.)

(b) Social Evolution Data Set (Number of iterations: Bluetooth: 7630, Phone: 8878, Text: 3144, WiFi: 4286, All: 12197.)

Fig. 2: Performance of Individual Features

As can be seen in Figures 2a and 2b, when we make use of all of the features in each data set, the performance is not better than the performance obtained with the best feature on its own.

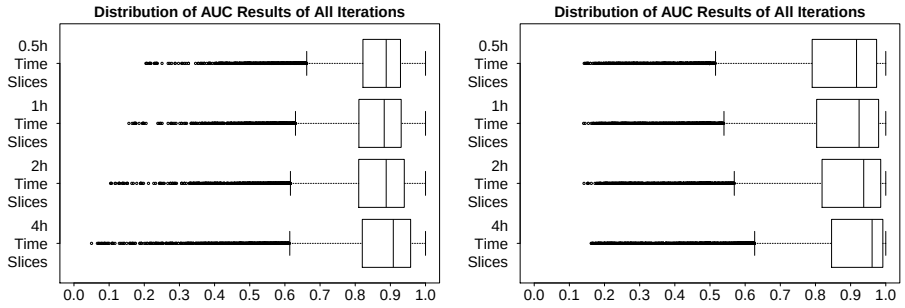
This observation motivated us to experiment with "match score level fusion," [17] where we trained one anomaly detection model per feature type and combined the anomaly scores of individual features of each test time slice to obtain one anomaly score per test time slice. We do not report our results with match score level fusion because this fusion method does not bring improvements over the method we have described in this paper.

Results with Different Time Slice Lengths In the above-mentioned experiments, we used a time slice length of 4 hours, which means that the anomaly detection delay of a mobile device can be up to 4 hours. We would like to reduce the time slice length in order to reduce the anomaly detection delay, but while doing so we also risk an increase in the time complexity of our method because shorter time slices translate to more time slices. The experiments in this section quantify the effects of reducing the time slice length from 4 hours to one half of an hour.

In these experiments, we use k-NN with k set to 1. We use all of the features, and we set the time slice leniency equal to 4 hours and set the training period length equal to 21 days.[11]

We present our results in Figures 3a and 3b, where we can observe that reducing the time slice length from 4 hours to one half of an hour reduces the median *and* the spread of the AUC values resulting from the experiments with

[11] We should also note that we use aggressive sampling with these experiments in order to keep the experiment run-times reasonable, which reduces the significance of the conclusions we can draw from the results of these experiments.



(a) Reality Mining Data Set (12431 iterations).

(b) Social Evolution Data Set (12197 iterations).

Fig. 3: Performance of Different Time Slice Lengths

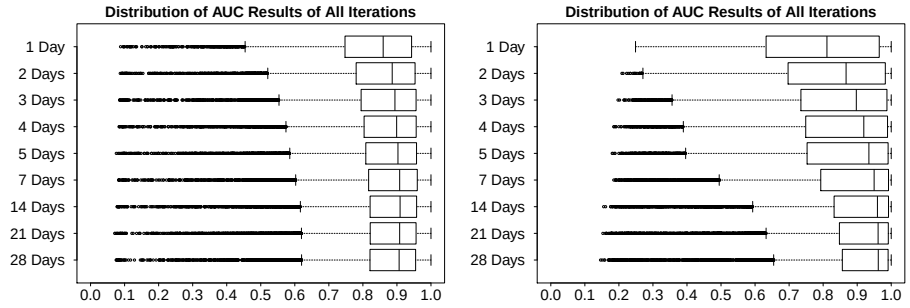
the Reality Mining data set, whereas with the Social Evolution data set the AUC value spread increases, and the median decreases. Despite these observations, we cannot conclusively state whether the reduction in the time slice length affects the performance of our method.

Results with Different Training Period Lengths Finally, we vary the training period length between 1 day and 28 days to observe the effect of the training period length on the accuracy of our approach. We would like to reduce the training period length as much as possible to reduce the time complexity of our approach, whereas reducing the training period length too much may degrade the accuracy of our method.

In these experiments, we use k-NN with k set to 1. We use all of the features, and we set the time slice length and leniency equal to 4 hours.

We present the results with the Reality Mining data set in Figure 4a, and those with the Social Evolution data set in Figure 4b. As can be seen in both figures, decreasing the training period length reduces the overall accuracy of our method, where the AUC value median decreases, and the AUC value spread increases. We can also observe that with the Social Evolution data set, the reduction in the overall accuracy is more visually apparent.

Variance in Classification Performance As can be seen in Figures 2a, 2b, 3a, 3b, 4a and 4b, the boxplots have relatively large spreads and long whiskers indicating high variances in our results. After plotting for each data set participant a scatter plot of the AUC values corresponding to each iteration/test day, we observe the following: (1) A number of data set participants' identity cannot reliably be verified via their mobile device usage patterns. These participants suffer from a large number of false positives (i.e. normal time slices predicted as anomalous) which highly vary the results across evaluation iterations. (2) Other data set participants whose mobile device usage patterns can be more reliably used for verification also suffer from occasional false positives.



(a) Reality Mining Data Set (b) Social Evolution Data Set

Fig. 4: Performance of Different Training Period Lengths (Each boxplot summarizes approximately 12000 iterations)

4.6 Mobile Device Feasibility Study

To show the feasibility of our method on mobile devices, we deploy our implementation on two actual smartphones. We should note that our intention with this deployment experiment is not the testing of our method's accuracy in a deployed setting.

Deployment Overview As the target deployment platforms, we choose Nokia N900, a smartphone from the year 2009, which has modest specifications in comparison to modern smartphones, and Samsung Galaxy Nexus, a more recent and more powerful smartphone from the year 2011.

We use Python programming language to implement and deploy our method.[12] Because Python is an interpreted programming language, we believe we incur a performance hit in the form of higher CPU utilization for computation-intensive tasks.

Experimental Set-up For this experiment, our software collects Bluetooth and WiFi discovery results every 25 and 30 seconds, respectively. We set the time slice length equal to one minute, and as a result perform a k-Nearest Neighbours-based anomaly detection run every minute against the training data, which is chosen to be the data from the past five days.

Note that the parameters of the aforementioned experimental set-up involve a departure from the parameters we use for the evaluation of our method's accuracy. We choose aggressive settings (such as frequent wireless discoveries and frequent anomaly detection iterations) to obtain a worst-case scenario in terms of the user experience and resource utilization.

[12] The N900's Linux-based operating system, Maemo, natively supports Python, whereas we resort to the Scripting Layer for Android (SL4A) to run our software on the Galaxy Nexus, which ships with Google's Linux-based Android operating system for mobile devices.

We evaluate the deployment qualitatively via a user experience study and quantitatively via resource utilization measurements. For the former, we report our findings resulting from the primary author's use of each of the target mobile devices for at least one week, during which the mobile device in question was used as a personal music player and to make occasional phone calls. For the latter, we report CPU, Random Access Memory (RAM) and persistent (Flash) memory utilization and the battery life.

Results Throughout the qualitative user experience study, from a computational performance point of view, we could not notice that our software had been running in the background. For example, we did not experience any interruptions in phone calls or music playback on either deployment platform, even though we could observe the CPU (via a "desktop" applet on the N900) utilization peak as anomaly detection runs were being performed every minute.

From an overall performance point of view, however, our software did have negative effects: On both deployed platforms, the battery life was affected negatively, and we needed to recharge the battery more frequently compared to what was needed without our anomaly detection software.

In quantitative terms, each anomaly detection run causes a maximum CPU utilization period lasting 6 to 7 seconds on the N900 and 2 to 3 seconds on the Galaxy Nexus. Memory usage, on both platforms, is between 75 and 80 MegaBytes with five days of training data loaded in memory. SQLite3 databases containing approximately 40 days of data use approximately 95 MegaBytes of persistent (Flash) memory.

Finally, battery life on the N900 is approximately 15–16 hours, whereas with the Galaxy Nexus the battery life is approximately 23–24 hours.

5 Conclusion

To conclude, in this work we use a time slice model to summarize contextual and behavioural information that can be obtained from some of the sensors found on modern smartphones and perform anomaly detection using a variant of the k-Nearest Neighbours algorithm.

We evaluate the accuracy of our method with the Reality Mining and the Social Evolution data sets. We find that location, phone call and Bluetooth discovery features to perform better than the other features of the Reality Mining data set, and that with the Social Evolution data set, phone call and Bluetooth discovery features to perform better than the other features.

We cannot conclusively state that the reduction of the time slice length from 4 hours to half an hour affects the accuracy of our method, and we verify that the length of the training period is positively correlated with accuracy.

Finally, we find that the impact of our method to a mobile device user's experience is acceptable in terms of computational performance, while we believe that the battery life can be improved by increasing the time slice length and the

feature collection periods at the cost of increased anomaly detection latency and possibly decreased accuracy.

As part of future work, we would like to evaluate the *accuracy* of our method on actual mobile devices, possibly with different time slice lengths. One of the preconditions to the evaluation of accuracy in deployed settings is the automatic calculation of the anomaly score threshold at run-time, for example by fixing the false positive rate to a certain percentage using the training data, as Yazji et al. do so in [4,5,9].

Personalized feature weighting when the mobile device retrains itself (which we envision would be performed while the device is being charged) is another future work direction.

Dissimilarity vector-based classification would allow us to adapt conventional machine learning algorithms – such as one-class SVMs or k-Means clustering – to our method and could potentially provide better accuracy.

Other data sets containing mobile device usage data, such as the Nodobo data set [18], can be used to further evaluate our method.

Last but not least, we would like to integrate our method with an implementation of the Device Comfort framework, where the anomaly scores produced by our method could be used, in part, to produce a trust (or comfort) level for the mobile device. Based on its comfort level, the user interface of a computing device may change its behaviour, for which there have been a number of proposals, which include but are not limited to those made by Storer et al. [19] and Murayama et al. [20].

Acknowledgements

This work was funded by the Communications Research Centre, Canada.

References

1. Marsh, S., Briggs, P., El-Khatib, K., Esfandiari, B., Stewart, J.A.: Defining and Investigating Device Comfort. Journal of Information Processing **19** (2011) 231–252
2. Shi, E., Niu, Y., Jakobsson, M., Chow, R.: Implicit authentication through learning user behavior. In Burmester, M., Tsudik, G., Magliveras, S., Ilic, I., eds.: Information Security. Volume 6531 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 99–113
3. Li, F.: Behaviour Profiling for Mobile Devices. PhD thesis, University of Plymouth, Plymouth, Devon, Great Britain, United Kingdom (2012)
4. Yazji, S., Dick, R.P., Scheuermann, P., Trajcevski, G.: Protecting private data on mobile systems based on spatio-temporal analysis. In Benavente-Peces, C., Filipe, J., eds.: PECCS, SciTePress (2011) 114–123
5. Yazji, S., Scheuermann, P., Dick, R., Trajcevski, G., Jin, R.: Efficient location aware intrusion detection to protect mobile devices. Personal and Ubiquitous Computing **18**(1) (2014) 143–162

6. Branscomb, A.S.: Behaviorally identifying smartphone users. Master's thesis, North Carolina State University, Raleigh, NC (2013)
7. Crawford, H.A.: A framework for continuous, transparent authentication on mobile devices. PhD thesis, University of Glasgow, Glasgow, Scotland, United Kingdom (2012)
8. Zhu, J., Wu, P., Wang, X., Zhang, J.: Sensec: Mobile security through passive sensing. In: Computing, Networking and Communications (ICNC), 2013 International Conference on. (Jan 2013) 1128–1133
9. Yazji, S., Chen, X., Dick, R., Scheuermann, P.: Implicit user re-authentication for mobile devices. In Zhang, D., Portmann, M., Tan, A.H., Indulska, J., eds.: Ubiquitous Intelligence and Computing. Volume 5585 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 325–339
10. Salem, M., Stolfo, S.: Modeling user search behavior for masquerade detection. In Sommer, R., Balzarotti, D., Maier, G., eds.: Recent Advances in Intrusion Detection. Volume 6961 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 181–200
11. Eagle, N., Pentland, A.S., Lazer, D.: Inferring friendship network structure by using mobile phone data. Proceedings of the National Academy of Sciences (2009)
12. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Engineering Bulletin **33**(2) (2010) 32–40
13. Madan, A., Cebrian, M., Moturu, S., Farrahi, K., Pentland, A.: Sensing the 'health state' of a community. Pervasive Computing, IEEE **11**(4) (Oct 2012) 36–45
14. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (July 2009) 15:1–15:58
15. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection. In Barbar, D., Jajodia, S., eds.: Applications of Data Mining in Computer Security. Volume 6 of Advances in Information Security. Springer US (2002) 77–101
16. Rawat, S.: Efficient Data Mining Algorithms for Intrusion Detection. PhD thesis, University of Hyderabad, Hyderabad, India (2005)
17. Ross, A., Jain, A.: Information fusion in biometrics. Pattern Recognition Letters **24**(13) (2003) 2115 – 2125 Audio- and Video-based Biometric Person Authentication (AVBPA 2001).
18. McDiarmid, A., Bell, S., Irvine, J., Banford, J.: Nodobo: Detailed mobile phone usage dataset. (2011) Unpublished. Available: `http://www.nodobo.com`.
19. Storer, T., Marsh, S., Noel, S., Esfandiari, B., El-Khatib, K., Briggs, P., Renaud, K., Bicakci, M.: Encouraging second thoughts: Obstructive user interfaces for raising security awareness. In: Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on. (July 2013) 366–368
20. Murayama, Y., Fujihara, Y., Saito, Y., Nishioka, D.: Usability issues in security. In Christianson, B., Malcolm, J., Stajano, F., Anderson, J., eds.: Security Protocols XX. Volume 7622 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 161–171